Content Placement Strategies for Smart Products

Zur Erlangung des akademischen Grades Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation von Dipl.-Wirt.-Inf. Markus Miche aus Wiesbaden
2015 — Darmstadt — D 17

TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Informatik
Telekooperation

Content Placement Strategies for Smart Products

Genehmigte Dissertation von Dipl.-Wirt.-Inf. Markus Miche aus Wiesbaden

1. Gutachten: Prof. Dr. Max Mühlhäuser
2. Gutachten: Prof. Dr. Antonio Krüger

Tag der Einreichung: 30.06.2014
Tag der Prüfung: 15.07.2014

Darmstadt — D 17

Danksagung

Der Weg von der Idee des Promovierens bis hin zur Fertigstellung der vorliegenden Arbeit wurde von vielen Menschen begleitet, ohne deren geschätzte Unterstützung ich mein Ziel kaum hätte erreichen können. An erster Stelle danke ich meinem Doktorvater Max Mühlhäuser. Du hast hat mir durch Deine kritische Sichtweise zu den richtigen Denkanstössen verholfen und mir die Freiräume und die Flexibilität eingeräumt, die für eine externe Promotion nötig sind. Auch den Kollegen des Lehrstuhls für "Telekooperation" möchte ich meinen Dank aussprechen. Ihr habt mir bei meinen wenigen Vorträgen wertvolles Feedback gegeben und hattet stets ein offenes Ohr für Fragen und Diskussionen.

Mein Dank gilt ferner den Mitarbeitern des Forschungsprojekts SmartProducts, insbesondere den Kollegen der Technischen Universität Darmstadt und der SAP Research. Ihr habt mich bei der Ausarbeitung meiner Doktorarbeit im Rahmen des Projekts begleitet und unterstützt. Auch den Kollegen von EADS Innovation Works bin ich aufgrund der gemeinsamen Definition und Modellierung des in der Arbeit genutzten Anwendungsszenarios zu grossem Dank verpflichtet. Hervorheben möchte ich weiterhin die Unterstützung von Oliver Kasten und Uli Eisert, sowie Oliver Bäcker und Volkmar Jäckle. Ihr habt mir die nötigen Freiräume gegeben, um meine Arbeit parallel zu meiner Anstellung bei SAP Research und SAP Consulting fertigstellen zu können. Ein besonderer Dank gilt ausserdem Kai Baumann. Kai, Du warst nicht nur während Deiner einjährigen Anstellung bei SAP Research sondern auch danach ein wertvoller Diskussionspartner. Vielen Dank für Dein Engagement und Deine Unterstützung.

Mein grösster Dank gilt meiner Frau Verena und meinem Sohn Paul sowie meinen Eltern Kathrin und Frank. Ihr standet mir immer zur Seite und wart mir insbesondere in schwierigen Zeiten die nötige Stütze. Ich danke Euch sehr für Euer Verständnis, Eure Geduld und Euer Einfühlungsvermögen. Ohne Euch hätte ich mein Ziel nicht erreichen können. Danke!

Ehrenwörtliche Erklärung[2]

Hiermit erkläre ich, die vorgelegte Arbeit zur Erlangung des akademischen Grades "Dr. Ing." mit dem Titel "Content Placement Strategies for Smart Products" selbständig und ausschließlich unter Verwendung der angegebenen Hilfsmittel erstellt zu haben. Ich habe bisher noch keinen Promotionsversuch unternommen.

Darmstadt, den 30.06.2014

Markus Miche

[2] Gemäß §9 Abs. 1 der Promotionsordnung der TU Darmstadt

Zusammenfassung

Technologischer Fortschritt und Standardisierungsaktivitäten sind die treibenden Kräfte einer stetigen Verbreitung sogenannter *smarter Dinge*. Das Spektrum smarter Dinge reicht von Objekten mit RFID-Transpondern bis hin zu komplexen Objekten mit integriertem Speicher sowie Rechen- und Kommunikationsfähigkeiten. Smarte Dinge ermöglichen sowohl neue Funktionalitäten und Anwendungen als auch Verbesserungen existierender Prozesse. Die vorliegende Arbeit befasst sich mit *smarten Produkten*, einer speziellen Klasse smarter Dinge die auf eine vereinfachte Produktnutzung mittels multimodaler und personalisierter Interaktionsstrategien abzielt. Dieses Verhalten wird durch modelliertes Wissen ermöglicht, welches mit smarten Produkten assoziiert ist und während des Produktlebenszyklus erlernt wird.

Smarte Produkte besitzen eingeschränkte Ressourcen und sehen sich typischerweise mit regelmässigen Verbindungsunterbrechungen konfrontiert. Folglich ist davon auszugehen, dass sie weder in der Lage sind alle benötigen Informationen lokal zu speichern, noch zu jeder Zeit eine Verbindung mit entfernten Systemen zwecks Erhalt / Ablage von Daten aufzubauen. Der Zugriff auf Informationen ist jedoch zentral für das Erreichen der Zielsetzung smarter Produkte. Dies zeigt die Notwendigkeit intelligenter Mechanismen, welche die Verfügbarkeit benötigter Informationen und einen effizienten Zugriff bestmöglich sicherstellen.

Die vorliegende Arbeit befasst sich mit der Fragestellung, auf welche Weise prozedurales Wissen smarter Produkte für eine Optimierung der Datenorganisation in verteilten Systemen genutzt werden kann. Gemäss der Ergebnisse des EU-geförderten Forschungsprojekts SmartProducts[3] nimmt die vorliegende Arbeit eine Modellierung prozeduralen Wissens in Form von Workflows an. Workflows stellen potentiell komplex verzweigte Sequenzen von Aktivitäten dar, welche mit den zur Prozessierung benötigen Daten annotiert sind.

Auf dieser Grundlage trägt die Arbeit drei Workflow-basierte Replikationsstrategien bei: *Most Probable Path (MPP)*, *Path Assessment (PA)* und *Cooperative Path Assessment (CPA)*. Die Strategien nutzen Workflowstrukturen um den zukünftigen Datenbedarf vorherzusagen und diese Daten pro-aktiv zu replizieren. Dies hat eine Verbesserung der Verfügbarkeit und Zugriffsgeschwindigkeit jener Daten zur Folge und ermöglicht eine effizientere Ausführung von Workflows. MPP, PA und CPA verfolgen identische Ziele, unterscheiden sich jedoch in der Balancierung der Verbesserung des Datenzugriffs und des potentiellen Risikos fehlerhafter Bedarfsprognosen.

Um das Risiko der Replikation nicht-benötigter Daten zu reduzieren, wenden die Workflow-basierten Replikationsstrategien das entwickelte Konzept *transienter Replikate* an. Transiente Replikate zeichnen sich durch eine beschränkte, mit jedem Zugriff

[3] `http://www.smartproducts-project.eu/`

steigende Lebenszeit aus. Bei einer ausreichenden Anzahl von Zugriffen werden transiente Replikate in persistente (dauerhaft gespeicherte) Replikate überführt. Das Ausbleiben eines Zugriffs innerhalb der geltenden Lebenszeit hat hingegen das Löschen des transienten Replikats zur Folge. Dies reduziert die Anzahl zu verwaltender Replikate und adressiert die Unsicherheit der pro-aktiven Replikation.

Des Weiteren ist die Replikationsstrategie *Content Class (CC)* vorgeschlagen. Im Gegensatz zu den Workflow-basierten Replikationsstrategien basiert CC nicht auf beobachteten und vorhergesagten Zugriffsmustern, sondern ermöglicht eine Klassifikation der geplanten Datennutzung durch Applikationen smarter Produkte. CC nutzt diese Klassifikation und stellt dedizierte Strategien zur Verfügung, welche die Organisation der Daten entsprechend der Datenklassifizierung bestmöglichst anpassen.

Die vorliegende Arbeit folgt der in der Forschungsgemeinde gängigen Auffassung, dass es gewinnbringend ist, Datenreplikations- und Datenaustauschstrategien integriert zu betrachten. Dementsprechend trägt die Arbeit zwei Datenaustauschstrategien bei: *MFR for Transient Replicas (MFRTR)* und *Enhanced Content Replacement (ECR)*. MFRTR komplementiert die Workflow-basierten Replikationsstrategien. Der Ansatz erweitert die bekannte Strategie MFR [KRT06, KRT07] um die Behandlung transienter Replikate. Beim Ausbleiben von Zugriffen auf transiente Replikate werden diese durch MFRTR nicht sofort gelöscht, sondern erst bei Auftreten von Anfragen deren Bedienung der Freisetzung zusätzlicher Speicherkapazität bedarf. Dies resultiert in einer effizienten Nutzung verfügbarer Speicherressourcen ohne die Aufwände des Datenaustauschs zu erhöhen. ECR erweitert MFRTR um die Nutzung von Datenklassifizierungen im Rahmen der Identifikation auszutauschender Daten. Auf diese Weise zielt ECR auf den Erhalt der Datenpositionierung entsprechend der Datenklassifizierung ab und vervollständigt die Strategie CC.

Der Mehrwert der genannten Strategien wurde anhand eines Simulationsmodells des Szenarios *Smart Aircraft Manufacturing*, welches gemeinsam mit EADS Innovation Works entwickelt wurde, evaluiert. Verglichen mit existierenden Replikationsstrategien zeigt die Studie eine Verbesserung der Datenzugriffsgeschwindigkeit um bis zu 28%. Reduziert auf Workflow-basierte Datenzugriffe steigt die Verbesserung auf bis zu 62%. Nach dem Wissen des Autors der vorliegenden Arbeit sind die vorgeschlagenen Strategien der erste Ansatz prozedurales Wissen smarter Produkte sowie Applikationsgetriebene Datenklassifizierungen für eine Optimierung der Datenorganisation in verteilten Systemen smarter Produkte einzusetzen.

Abstract

Driven by advances in technology as well as standardisation efforts, the adoption of *smart things* gathers pace in various industries. Smart things range from simple objects being equipped with smart labels to comprehensive objects with embedded storage, computing, and networking capabilities. They aim at improving product operation and usage as well as at enabling functionality beyond their original purpose of use. Think of connected cars that seek for enhancing traffic safety and efficiency as well as for enabling telematic services, or industrial assets that sense and monitor their condition in order to improve maintenance operations. This thesis addresses *smart products*, a specific class of smart things that targets simplicity of product use by means of multi-modal and personalised product-to-user interaction. For this purpose, smart products are equipped with knowledge and knowledge-related functionality. Amongst others, this includes domain and problem-solving knowledge that is used to enable active user guidance.

Given their resource limitation and potential intermittent connectivity, smart products are typically not able to locally store all content required and created across their life cycle neither can they connect to remote systems at all times for storing and / or retrieving information. Yet, in order to achieve simplicity of product use, it is essential that product-to-user interaction is not distorted by long user-perceived delays for retrieving required information or by falling back to simple interaction means, because of content being not accessible. Hence, there is a need for intelligent mechanisms that make content available when required and accessible with low latency.

This thesis studies how procedural problem-solving knowledge associated with smart products can be utilised by content placement strategies, i.e., content replication and replacement strategies, to enhance content access. According to the results of the EU-funded research project SmartProducts[4], the thesis assumes procedural problem-solving knowledge to be modelled in the form of workflows that reflect sequences of activities and feature annotation of activity-related content needs. Also, it assumes complex-structured workflows that consist of multiple alternative branches with the actual branch(es) to be followed being determined dynamically.

On this basis, the thesis contributes three workflow-based replication strategies, namely *Most Probable Path (MPP)*, *Path Assessment (PA)*, and *Cooperative Path Assessment (CPA)*. These strategies utilise workflow structures to predict and pre-replicate (i.e., replicate in advance) workflow-related content needs in order to enhance query efficiency of content requests during workflow execution. While MPP, PA, and CPA pursue the same objective, they vary in the way of balancing enhancement of query efficiency with the potential risk of false pre-replication as well as in their incorporated sphere of knowledge.

[4] http://www.smartproducts-project.eu/

In order to address uncertainty in execution of branched workflows and the resulting risk of falsely pre-replicated content, the workflow-based replication strategies apply the proposed concept of *transient replicas*. Transient replicas are assigned gradually increasing Time To Live (TTL) intervals. While they may be persisted in case the number of requests exceeds a pre-defined threshold, they are removed in case they are not accessed within their active TTL. This avoids pre-replication wastage in the long term and limits the number of replicas to be maintained.

Moreover, the replication strategy *Content Class (CC)* is proposed. Instead of relying on observed or predicted access patterns, this approach enables smart product applications to classify their intended content use and provides content-class-specific best effort policies to improve content organisation.

Given the typical benefits of integrated content replication and replacement strategies, this thesis proposes the two replacement strategies *MFR for Transient Replicas (MFRTR)* and *Enhanced Content Replacement (ECR)* that complement the workflow-based replication strategies and CC, respectively. MFRTR extends the well-known replacement policy MFR [KRT06, KRT07] by taking into account transient replicas with lazy removal properties. Thus, transient replicas are only removed if additional storage capacity is needed for serving pending storage operations. This results in efficient utilisation of available storage capacity without increasing replacement overhead. ECR extends MFRTR by leveraging content classification for determining replacement candidates. Instead of purely relying on access frequency and recency, this approach aims at preserving location properties of content according to its classification.

The value of the proposed content placement strategies is assessed with a simulation-based evaluation study that adopts the application scenario *smart aircraft manufacturing* developed jointly with EADS Innovation Works. This study reveals that the proposed strategies improve overall and workflow-related query efficiency by up to 28% and 62%, respectively, compared to related work. As to the knowledge of the author, the proposed content placement strategies are the first that leverage procedural problem-solving knowledge associated with smart products and enable application-affected content organisation while taking into account the challenges of smart products.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

ACID	Atomicity, Consistency, Isolation, and Durability
API	Application Programming Interface
AS	Autonomous System
BASE	Basically Available, Soft-state, Eventual consistency
CAP	Consistency, Availability, Partition tolerance
CAPI	Common API
CDN	Content Distribution Network
CQORP	Cost-Quality Optimised Replica Placement
DHT	Distributed Hash Table
DOM	Digital Object Memory
FMC	Fundamental Modelling Concepts
HTL	Hops To Live
IRM	Integrated file Replication and consistency Maintenance
IT	Information Technology
LFU	Least Frequently Used
LRU	Least Recently Used
MDCDN	Mobile Dynamic Content Distribution Network
MFR	Most Frequently Requested
MRU	Most Recently Used
P2P	Peer-to-Peer
PHFS	Predictive Hierarchical Fast Spread
PWS	Predictive Working Set
QoS	Quality of Service
RTT	Round Trip Time
SLA	Service Level Agreement
SPA	Sequence Prediction Algorithm
TTL	Time To Live
UML	Unified Modelling Language
W3C	World Wide Web Consortium
WfMC	Workflow Management Coalition
XPDL4USE	XPDL for Ubiquitous Semantic Environments

Chapter 1
Introduction

1.1 Background

Driven by speedy technological progress, more and more products are equipped with computing and networking resources in order to improve product operation and usage as well as to enable functionality beyond their original purpose of use. Think of television sets with embedded application platforms, connected cars that seek for enhancing traffic safety and efficiency as well as for enabling telematic services, or industrial assets that sense and monitor their condition in order to improve maintenance operations. Despite the potential of this trend, it entails growth of complexity and diversity of technical products that is fostered by configuration options and shortened product life cycles. Users tend to be challenged and overwhelmed by the increasing functions and features of technical products as well as the amount of associated information [Res09].

This thesis targets *smart products*. Informal speaking, smart products are tangible objects with embedded computing and networking resources that aim at simplifying product use by means of personalised product-to-user interaction and active user guidance (a detailed definition of smart products is presented in Section 2.2). Instead of having the user understand and execute product functionality, it is the smart product that infers situational knowledge and user objectives in order to approach and assist the latter in a proactive way. For this purpose, smart products are associated with knowledge and knowledge-related functionality that allow for autonomous and cooperative task accomplishment. Knowledge associated with smart products can be classified into (i) *procedural knowledge* and (ii) *reasoning-based knowledge* that consists of tools and methods for dynamic process composition, task assignment, and problem solving. While procedural knowledge is often used for well-defined processes such as assembly or product maintenance procedures, the inherent flexibility of reasoning-based knowledge copes for cooperative problem solving in dynamically orchestrated smart product systems [NdGV11].

In addition to collaboration for improved simplicity and “smartness”, smart products share resources and capabilities to address the typical limitation of their embedded resources. Amongst others, this encompasses usage of interaction devices that are available in the environment to enable different interaction modalities as well as collaborative storage mechanisms in order to make available information required for user guidance and to improve access efficiency.

1.2 Motivation

The realisation of smart products poses challenges in various research domains. Amongst others, these challenges include mechanisms for embedding smart products in different, dynamically orchestrated environments across their life cycle, ontologies for describing domain and problem-solving knowledge, as well as means for enabling natural product-to-user interaction. Also, organisation and management of product- and user-related content have to be addressed. Think of a smart product being shared by multiple users who operate the product in different environments. Such a product must either store or dynamically retrieve user interface options and related media depending on the user, the intended use of the product, and the interaction resources of the environment.

Given the typically limited resources of smart products, it cannot be assumed that smart products are capable of storing all content required in the course of their life cycle on-board. On-demand retrieval of content provided by remote storage resources – however – results in increased *access latency* in almost all cases. Moreover, due to heterogeneous communication technologies and their potential mobility, smart products may face regular disconnections. Indeed, *connectivity* to remote storage resources cannot be assumed in all environments smart products may be operating in. While connectivity is generally available in well-equipped manufacturing scenarios, domains such as mobile field services or logistics are characterised by occasional connectivity. In conclusion, smart products are typically not able to locally store all content required and created across their life cycle neither can they connect to remote systems at all times for storing and / or retrieving information.

In order to achieve simplicity of product use, it is essential that product-to-user interaction is not distorted by long user-perceived delays for retrieving required information or by falling back to simple interaction means, because of content being not accessible. Hence, there is a need for intelligent mechanisms that make content available when required and accessible with low latency taking into consideration the challenges of smart products.

1.3 Research Question

This thesis studies means for improving availability and query efficiency of content required by smart products given the above-stated challenges. In particular, it analysis how knowledge associated with smart products can be utilised by content placement strategies for enhancing content access. This objective is subsumed in the research question presented below, which applies the *3 step formula* proposed by [BCW08].

1. **Topic.** *I am working on* content placement strategies for smart products,
2. **Conceptual Question.** *because I want to find out* how knowledge associated with smart products can be utilised for improving query efficiency and content availability in distributed smart product systems,
3. **Significance.** *so that readers can* build distributed storage systems for smart products that improve content access taking into account that smart products may neither be able to store all content required and created across their life cycle on-board nor to connect to remote storage resources at all times.

1.4 Contributions

The thesis contributes a set of content placement strategies for smart products, which consists of content replication as well as complementary content replacement policies. The novel strategies make use of procedural knowledge associated with smart products in order to enhance availability and query efficiency of content needed by smart products.

The strategies were developed by the author in the EU-funded research project SmartProducts[5] and build upon the project's results. The SmartProducts project models procedural knowledge associated with smart products as workflows that reflect sequences of activities. Workflows are assumed to be annotated with activity-related content needs such as user interface options and are used to model product operations as well as to enable active user guidance. Moreover, research activities in the SmartProducts project revealed that workflows in the domain of smart products often feature multiple alternative branches with the actual branch(es) to be followed being determined dynamically during runtime. A detailed description of workflows and workflow-based operations is presented in Section 2.1.5.

[5] `http://www.smartproducts-project.eu/`

The potential of workflow-aware distributed storage mechanisms for enhancing content access and – as a result – workflow execution performance is stated by [VAKC+12]. Given the typical content access patterns resulting from workflow execution, distributed storage mechanisms are capable of enhancing content placement by making use of either (i) explicit hints on activity-related content needs provided by a workflow engine, or (ii) knowledge inferred by the distributed storage mechanism based on past workflow executions.

The thesis proposes three workflow-based replication strategies, namely *Most Probable Path (MPP)*, *Path Assessment (PA)*, and *Cooperative Path Assessment (CPA)* that combine the two above-stated concepts. They utilise workflow structures to predict and pre-replicate (i.e., replicate in advance) upcoming workflow-related content needs in order to enhance query efficiency of content requests during workflow execution. Given the structural complexity of workflows, the strategies address the questions of (i) when to process pre-replication, (ii) for how many activities to pre-replicate in advance, (iii) how to handle uncertainties in branched workflows, and (iv) how to deal with falsely pre-replicated content. While MPP, PA, and CPA pursue the same objective, they vary in the way of balancing enhancement of query efficiency with the potential risk of false pre-replication.

Moreover, the replication strategy *Content Class (CC)* is proposed that targets optimisation of content access independent of workflow operations. CC enables content classification and provides content-class-specific policies to adjust number and placement of replicas.

Finally, the thesis proposes two content replacement strategies *Most Frequently Requested for Transient Replicas (MFRTR)* and *Enhanced Content Replacement (ECR)* that complement the workflow-based and content-class-driven strategies, respectively.[6]

Note that the proposed content placement strategies are based on two main simplifications. First, the strategies aim at enhancing availability and query efficiency of immutable content. Even though there is a strong interrelation between content placement and consistency maintenance strategies, the latter are out of scope of this thesis. Second, the strategies focus on procedural knowledge of smart products modelled in the form of workflows, only. In order to realise a complete distributed storage system for the domain of smart products, future work should complement the proposed content placement strategies by (i) tailored consistency maintenance concepts that balance the trade-off between content availability / query efficiency and content consistency as well as (ii) concepts for enhancing content availability and query efficiency based on reasoning-based knowledge associated with smart products.

[6] The ordering in which the proposed strategies are presented reflects their weighting. Hence, the workflow-based replication strategies are the main contribution of this thesis. The replacement concepts represent additions to these strategies in order to reflect the complete replication life cycle.

1.5 Thesis Organisation

The thesis is organised as follows. Chapter 2 presents background information and introduces the terminology used in the scope of this thesis. Amongst others, this includes content placement strategies, the content properties content availability and query efficiency used in the research question, as well as workflows and workflow-based operations. In addition, the domain of smart products is introduced and the concepts developed in the course of the SmartProducts project are summarised. To exemplify the use of smart products, the application scenario "Smart Aircraft Manufacturing" is presented. The scenario was jointly developed with EADS Innovation Works and is used as reference scenario both for pointing out the application of the proposed mechanisms as well as for evaluating the latter. Furthermore, the roles smart products may play in a distributed storage system in general as well as during the exchange of related messages are defined. Finally, this chapter derives the requirements to be addressed by content placement strategies to enhance content access in smart product systems.

Chapter 3 presents the results of a study on state-of-the-art content placement strategies. Also, a novel classification scheme is proposed and used for categorising related work. Each strategy is summarised and the extent to which it supports the investigated requirements is analysed and discussed.

In Chapter 4, the major novel contributions of the thesis are presented. Moreover, the chapter discusses different operating models of the proposed strategies taking into account the resource limitation of smart products. The strategies are classified using the above-stated scheme and mapped with the requirements investigated in Chapter 2.

In Chapter 5, the impact of the proposed content placement strategies on content availability and query efficiency is assessed by means of a simulation-based evaluation study. The chapter adopts the iterative modelling and simulation process proposed by [Kö01] and presents the conceptual simulation model in Section 5.2, the implemented simulation model in Section 5.3, and the actual simulation data and evaluation results in Section 5.4. The latter details the configuration of the proposed strategies, presents an analysis of the simulation data, and gives recommendations of when to use which of the proposed strategies.

The thesis concludes in Chapter 6 with a summary of the results and an outlook on potential future work.

Chapter 2
Background, Definitions, and Requirements

This chapter provides background information and introduces the terminology used in the scope of this thesis to ensure an unambiguous understanding of the proposed content placement strategies. Moreover, the requirements to be addressed by content placement strategies targeting the goals presented in Section 1.2 in the domain of smart products are presented and discussed.

The chapter is structured as follows. Section 2.1 presents and discusses concepts the novel content placement strategies are based upon. Also, different terms and definitions used in literature are discussed and the variant used in the scope of this thesis is highlighted. Section 2.1.1 provides a high-level description of distributed storage systems along a set of design decisions that are relevant for framing the proposed strategies. Content placement strategies covering both replication and replacement functionality are discussed in Section 2.1.2. Also, the content properties availability, persistence, and query efficiency used for assessing the proposed strategies within the evaluation study are explained (see Section 2.1.3). Peer-to-Peer (P2P) overlay networks are used in many of the prototyped smart product application scenarios [AKM07] and are moreover incorporated in the evaluation study of the thesis. For this reason, Section 2.1.4 provides an overview of P2P overlay networks. Finally, given the three proposed workflow-based replication strategies, workflows and workflow-based operations are discussed in Section 2.1.5.

Section 2.2 introduces the domain of smart products. This covers a definition of smart products as well as their challenges in Section 2.2.1. Section 2.2.2 presents the conceptual basis for realising smart products. According to the proposed content placement strategies, a special focus is set on workflow-based operations (i.e., procedural knowledge) as well as distributed storage concepts used in smart product systems. Section 2.2.3 presents the application scenario *smart aircraft manufacturing*, which is used as reference scenario to exemplify the use of smart products. Additional scenarios in the consumer electronics and automotive industry are presented in [KMS+12, MEA+11].

Thereafter, Section 2.3 introduces the terminology used in the description of the proposed content placement strategies. This includes the roles smart products may play in a distributed storage system in general as well as during the exchange of related messages.

Finally, the requirements for content placement strategies aiming at achieving the goals presented in Section 1.2 given the challenges of smart products are discussed in Section 2.4. These requirements are derived from the concepts of smart products as well as the exemplary application scenario presented in Sections 2.2.2 and 2.2.3, respectively. The requirements are addressed by the proposed content placement strategies and are moreover used to compare the latter with related work. Section 2.5 closes the chapter with a short summary.

This chapter is based upon and partly reuses descriptions out of the following publications of the author of this thesis: [MEA+11, MSH09, MKH11, MSB11, MBGB11]. Note that paragraphs reused from these publications are not marked as direct quotes if they were written exclusively by the author of this thesis. Quotations within reused paragraphs are taken over and marked accordingly.

2.1 Background

2.1.1 Distributed Storage Systems

Distributed storage systems can be characterised and classified on the basis of a set of design decisions such as the number of writers, the applied replication and consistency maintenance strategy, or the network architecture they are operating upon. The actual design of a distributed storage system is highly affected by the primary goals it has to address. Examples include strong consistency, i.e., users perceive the system as a single database, and distributed storage systems supporting high levels of query efficiency or content availability. A simplified overview of a subset of design decisions of distributed storage systems is presented in Figure 2.1 [MPV06, SS05, GHOS96].

Number of Writers. One central design decision of distributed storage systems is the supported number of writers. Thus, one can distinguish systems supporting multiple nodes to perform write operations (*multi-master system* with update anywhere properties, also known as group ownership) from systems that limit write operations to a single primary node per content while other nodes are only capable of performing read operations (*single-master system*, also known as master / slave ownership). While single-master systems avoid conflicting versions of content objects, they generally limit content

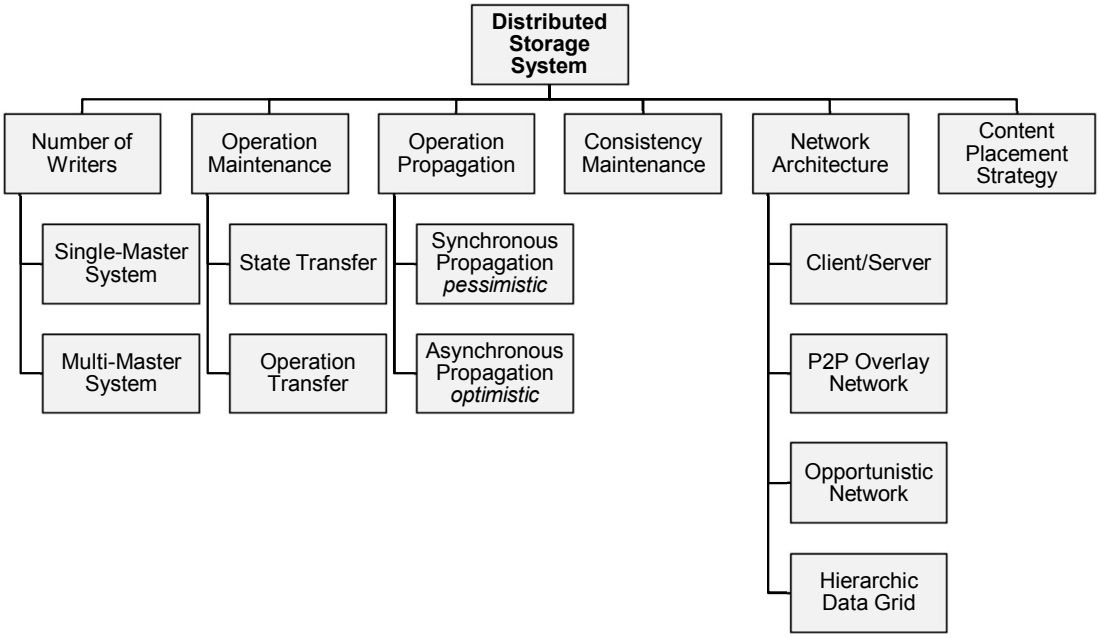


Figure 2.1: Design Decisions of Distributed Storage Systems

availability in case of failures of the primary node. Yet, multi-master systems imply the need for complex consistency maintenance strategies [MPV06, GHOS96].

Operation Maintenance. The way of documenting write operations in order to maintain consistency of distributed content objects is another central design decision. On the one hand, *operation-transfer systems* document semantic descriptions of operations (persistent operations). *State-transfer systems*, on the other hand, do not persist operations and solely maintain version information by means of – for example – timestamps (transient operations, operations are discarded after having been applied) [MPV06]. Means of documenting content updates that maintain happens-before relationship and causality between operations are, amongst others, vector clocks [Mat89], logical (Lamport) clocks, and real-time clocks [SS05]. While the state-transfer approach is typically less complex, it has the drawback of supporting only restricted conflict detection and resolution functionality. In contrast, the operation-transfer approach enables efficient conflict management functionality that utilises knowledge about operation semantics.

Operation Propagation. The actual propagation of either entire content objects (state transfer) or the corresponding set of operations (operation transfer) represents a design decision that strongly affects quality and performance of distributed storage systems. The Consistency, Availability, Partition tolerance (CAP) theorem invented by [Bre00] states that distributed storage systems are only capable of fulfilling two of the three CAP properties – consistency, availability, and partition tolerance. To support varying

network conditions, the trade-off typically remains between content consistency and content availability. The two known extremes of this trade-off are Atomicity, Consistency, Isolation, and Durability (ACID) systems on the one hand, which focus on strong consistency and *synchronous propagation* of content updates (also known as eager propagation [GHOS96]), and Basically Available, Soft-state, Eventual consistency (BASE) systems on the other hand, which relax consistency by allowing for *asynchronous propagation* policies (also known as lazy propagation [GHOS96]). By weakening consistency requirements, optimistic BASE systems achieve not only enhanced availability but further lead to significant scalability improvements [Pri08]. Moreover, BASE properties support disconnected write operations and can be configured to control the actual level of object divergence [Vog08]. Of course, this implies the risk of stale content versions and requires sophisticated consistency maintenance functionality. An overview of design decisions of consistency maintenance strategies with different consistency guarantees as well as a study on related work is presented in [MEA+11].

Network Architecture. Furthermore, distributed storage systems have to be designed according to the network architecture they are operating upon. Amongst others, this includes traditional *client / server* architectures (e.g., GFS [GGL03]), logical *P2P overlay network* structures (e.g., OceanStore [KWZ+00], PAST [RD01b]), *hierarchical structures* typically used in data grids (e.g., survey provided by [ASD12]), and *opportunistic networks* [Hei07].

Finally, content placement strategies are employed by distributed storage systems to enable content distribution as well as to maintain content locality given system dynamics. Due to the scope of this thesis, content placement strategies are detailed separately in the following section.

2.1.2 Content Placement Strategies

This thesis uses the term *content placement strategy* to cover both replication and replacement functionality. Content placement strategies utilise experienced and / or estimated access patterns (i.e., the distribution of content requests) in order to improve query efficiency, content availability, and / or content persistence (an explanation of these terms is presented in Section 2.1.3). This section presents informal definitions of the terms *content replication* and *content replacement* used in the scope of this thesis, and discusses usage of the two terms in related work.

Content Replication. Content replication strategies deal with the creation and distribution of copies of content – so called *replicas* – in order to enhance content access.[7]

[7] Note that in this thesis, the term *content* is used as an umbrella term covering content as well as replicas. If content is used in its exclusive definition (i.e., content $\neq$ replica), then this is explicitly stated to avoid ambiguity.

In general, query efficiency, content availability, as well as content persistence could be optimised by placing replicas of all content on all nodes of a distributed storage system. However, because of system constraints such as storage capacity of nodes or consistency maintenance overhead implied by high replication degrees, this approach is typically not applicable and there is a need for more intelligent replication strategies. Simply stated, content replication strategies have to address the question of *when* to place *how many* replicas *where*, i.e., on which nodes of the distributed storage system, as well as how to *maintain* number and placement of replicas given system dynamics [ASD12].[8]

Reactive Content Replication. Replication decisions are typically made based on observations of past requests and assume access patterns with *temporal locality* [RF01] properties. Temporal locality properties are present if content being requested is likely to be accessed again in the nearby future. Hence, content replication strategies typically operate in a *reactive* way and aim at improving expected upcoming content access.

In order to reduce the number of replicas in distributed storage systems, replication decisions are often made by means of *threshold-based policies*. Hence, a replica is not made directly after the first access but when the request rate exceeds a pre-defined threshold in a given period of time. This typically results in more informed replication decisions, because of the increased probability of access patterns following temporal locality properties.

Moreover, replication decisions can be made by assuming *geographic locality* properties [RF01], which take into consideration content interest of nodes within a certain geographic region. Simply speaking, these approaches assume that content being requested in a certain geographic region is likely to be accessed again in the nearby future by nodes that reside within the same region.[9]

Content Caching. Often, in related work, the term content replication is mixed with the term content caching. In fact, these terms are not mutually exclusive and show overlapping – especially if caching is combined with threshold-based policies. The dictionary Merriam Webster defines the term cache as "a computer memory with very short access time used for storage of *frequently or recently used* instructions or *data*"[10]. According to this definition, caching also relies on access patterns with temporal locality properties to decide on which content to be cached.

[8] In related work, replication strategies and consistency maintenance strategies are often mixed and subsumed as replication strategies [MPV06, BF11]. This thesis strictly separates replication from consistency maintenance in order to scope the proposed content placement strategies.

[9] [RF01] moreover defines access patterns with *spatial locality* properties. This means, content "nearby" recently accessed content is likely to be requested in the nearby future. [JIR+09] specifies "nearby" with geographic closeness in the domain of location-based services. Semantic relationship between content represents another potential specification of spatial locality and is often utilised by strategies that apply data mining algorithms [KIS11].

[10] `http://www.merriam-webster.com/dictionary/cache`

To clearly distinguish caching and replication, this thesis understands caching as the creation of a replica on-board of the requester after a successful retrieval of the requested content. Hence, caching strategies equal reactive replication strategies that do not apply threshold-based policies.

Active Content Replication. In addition to traditional reactive replication strategies, there are *active* strategies that base their decisions on *explicit predictions* of upcoming content demand and system state. Examples for such content access predictors include statistical demand forecasting based on access histories or location-based strategies that predict future content requests based on the location at which nodes are expected to be used in upcoming periods.[11] While these strategies are known to greatly improve content availability and query efficiency, they bear the risk of false replication, i.e., replicated content not being requested as expected [PK08].

In literature, different synonymous terms are used to describe active content replication. Most often, the terms "pre-fetching" [BWYH06, SGAM11], "pre-staging" [Fra01, CDL+07], and "pre-allocation" [PJFY04] are used for denoting active replication and content placement. This thesis adopts the term *pre-replication* used by [KIS11, ASD12] to describe replication strategies that actively create and place replicas based on predicted upcoming demand. Details about prediction algorithms, the risk of pre-replication wastage, and an analysis of the trade-off between the latter and enhancements of content access are provided in the subsequent sections.

Content Replacement. Content replacement strategies target management of the limited storage resources of nodes in distributed storage systems.[12] Replacement strategies are typically applied in case of pending storage requests that can be served only after removing other content from the node's storage. In related work, replacement strategies are also referred to as "cache algorithms" such as Most Recently Used (MRU) or Least Recently Used (LRU) [Smi82].

In general, two types of replacement strategies can be distinguished. On the one hand, especially in traditional caching scenarios, content replacement strategies *delete* content for serving pending storage requests. On the other hand, there are approaches that select content for being *moved* to other nodes.[13] While moving content to other nodes is indeed

[11] In fact, reactive strategies assuming access patterns with temporal or geographic locality properties also make predictions of upcoming demand. Yet, this form of *implicit* prediction is distinguished from active replication strategies that apply prediction algorithms to *explicitly* forecast future requests.

[12] Typically, the storage capacity of a node is logically partitioned into a private and a shared storage area. While private storage is in general application-managed, shared storage is made available for the distributed storage system and is managed by the latter in a (semi-) transparent way. Yet, for reasons of simplicity, this differentiation is not made in this thesis. Certainly, this does not affect the applicability of the proposed content placement strategies by any means.

[13] Note that replacement strategies typically focus on the determination of replacement candidates, only. The actual placement of these candidates is taken care of by other components of the related distributed storage system.

more complex, it has lower impact on content availability and query efficiency compared to replacement strategies deleting content out of the storage of nodes.

Hence, content placement strategies subsuming content replication and content replacement addresses all phases of the replication life cycle defined by [ASD12]: "replica creation, placement, relocation [i.e., replacement] and retirement [i.e., removal]" [ASD12, p. 8]. A detailed discussion and analysis of design decisions of content placement strategies as well as the outcome of a study on related work are presented in Chapter 3.

2.1.3 Content Properties in Distributed Storage Systems

This section shortly explains content properties relevant with respect to the overall goals presented in Section 1.2. This includes content availability, query efficiency, as well as content persistence. A definition of the properties used for evaluating the proposed content placement strategies is presented in Section 5.2.4.

Note that there are many more content-related properties in distributed storage systems such as security and privacy as well as content consistency. Yet, these are out of scope of this thesis. A detailed description of security and privacy as well as content consistency in distributed storage systems is presented in [Bec11] and [MEA+11], respectively.

Content availability. This thesis uses the term content available to denote the probability that at least one node holding a requested content is accessible by the requesting node given potential restrictions such as timeout properties associated with the request. Hence, content availability is affected by the storage location of the requested content (is the storage location accessible by the requesting node) and the replication degree of the requested content (the higher the replication degree, the higher the probability of at least one storage location being accessible by the requesting node). This understanding strictly differentiates content availability from query efficiency (see below), which is often mixed and assessed jointly [KIS11].

Query efficiency. In this thesis the term query efficiency is understood as the time needed for serving a content request in a distributed storage system. This relates to the time interval between a content request and the retrieval of the associated response, and is often referred to as the Round Trip Time (RTT) of content requests (see Section 5.2.4.3). *Access latency* is an indicator for query efficiency. Note that given the objectives of the proposed content placement strategies and the applied evaluation metric, this thesis uses both terms interchangeably (see also [HA04]).

Content persistence. This thesis understands content persistence from the perspective of content storage operations. Persistence denotes that content is never lost given system dynamics even though it might not be availability at all times. Hence, availability

implies persistence whereas the opposite does not need to be always given [DBEN07]. While transient failures or transient disconnections of nodes in distributed storage systems may affect content availability, permanently failing or leaving nodes may affect content persistence.

2.1.4 P2P Overlay Networks

P2P overlay networks are used in many of the prototyped smart product application scenarios [AKM07] and are moreover incorporated in the evaluation study of the thesis. To ensure a common understanding of the related terminology, which is in particular used in Chapter 5, this section presents a simplified classification of P2P overlay networks as well as an overview of related work.

In recent years, P2P technologies have become a very popular alternative to the traditional client / server approach. Especially in large-scale and dynamic systems, the distributed nature and self-organisation capabilities of P2P systems constitute promising means for approaching scalability challenges. P2P systems are based on overlay networks, which organise nodes in a logical network topology on top of the underlying network (e.g., IP network). Overlay networks can be classified into *structured overlay networks* (often also referred to as Distributed Hash Table (DHT)), in which topology and content placement follows specific rules, and *unstructured overlay networks*, in which nodes build an arbitrary structure and content can be placed anywhere. This also affects content lookup. While unstructured overlay networks require P2P systems to search for content (e.g., using constrained flooding or random walk techniques), content can be addressed in structured overlay networks according to the rules used to manage the overlay network topology [Bro10, LCP+05].

Structured Overlay Networks. The main challenge for structured overlay networks is the maintenance of a structure that enables efficient routing. This includes mechanisms that capture node churn (i.e., nodes joining and leaving the network) as well as the actual routing procedure such as the well-known prefix matching approach proposed by Plaxton [PRR99]. Typically, both nodes and content objects are assigned unique identifiers that can be calculated for example as a hash function of node properties (e.g., IP address) or content metadata. Based on their identifier, nodes are organised in a logical space (e.g., hash space) and are responsible for content with identifiers being numerically close to their node identifier, i.e., objects that are mapped near to them in the logical space. Well-known content location and routing substrates for managing structured overlay networks include Pastry [RD01a] and Tapestry [ZKJ01, ZHS+04], which both apply Plaxton's method. Further approaches are Chord proposed by [SMK+01] as well as CAN [RFH+01], which uses a virtual d-dimensional coordinate space for content placement and routing purposes.

Unstructured Overlay Networks. Unstructured overlay networks can be further classified as centralised, fully distributed, and hybrid approaches depending on the processing of search requests. *Centralised unstructured overlay networks* include a single central node that maintains a searchable index of all content shared by the nodes of the P2P system. While this enables very efficient searching, the central node is a single point of failure and might limit system scalability.

In *fully distributed unstructured overlay networks*, all nodes play the same role. This results in a very robust network structure. However, searching is rather inefficient as only basic mechanisms such as flooding can be applied.

Finally, *hybrid unstructured overlay networks* combine properties of centralised and distributed architectures. Nodes are divided into a set of clusters with each cluster being organised by a so-called super-node that plays the role of a local central entity and manages a number of assigned non-super-nodes, which are typically referred to as ordinary-nodes. Similar to the central entity in centralised unstructured overlay networks, super-nodes maintain a searchable index of content available within their cluster (i.e., content stored on-board of their children). Moreover, super-nodes may facilitate consistency maintenance processes and may be used by ordinary-nodes as proxies for enhanced content lookup and retrieval. Hybrid overlay networks take into account the inherent heterogeneity of nodes given in most P2P systems, enable efficient searching, and achieve high levels of scalability.

Heterogeneity in Structured Overlay Networks. While traditional hybrid approaches such as Gnutella v0.6 [KM02] organise super-nodes in arbitrary structures (i.e., unstructured overlay networks), recent work also proposes to use structured overlay networks to interconnect super-nodes. Examples include SuperPastry [CMR05], which adapts Pastry to support the hybrid structure of Gnutella v0.6, and the approach proposed by [LXL08] that defines a hybrid overlay network based on Chord. While this results in additional overhead for maintaining the structured overlay of super-nodes, it facilitates very efficient content location and routing. Hence, structured hybrid overlay networks are in particular suited for systems that include stable and reliable nodes that can play the role of super-nodes [Kan07].

Also, heterogeneity can be reflected in fully distributed overlay networks. The P2P file sharing system Gia [CRBS03] employs a capacity-aware topology so that nodes with higher capacity receive and handle more traffic. For this purpose, Gia considers bandwidth contributed by nodes as well as node uptime (i.e., the time they participate in the overlay network). Similar to SuperPastry referred to above, HeteroPastry [CMR05] is an extension of Pastry that applies the topology adaptation of Gia. According to the evaluation results presented in [CMR05], this results in improved scalability and a high level of query efficiency.

2.1.5 Workflow-based Operations

Business Processes & Workflows. *Business processes* are commonly defined as a set of logically interrelated activities that are structured with the intention of serving a particular organisational or commercial objective. Business processes reflect the business-level of a certain operation and are mainly used for communication and illustration purposes. Hence, they represent the high-level structure and flow of an operation and are typically non-exhaustive with respect to the execution on or configuration of process-aware information systems [Coa99, MSMP11, DVDA04].

The technical level of a business processes details execution semantics needed for process enactment. The Workflow Management Coalition (WfMC) defines *workflows* as "the automation of a business process, in whole or part, [..] according to a set of procedural rules" [Coa99, p. 8]. Hence, workflows represent Information Technology (IT)-supported implementations of business processes [Rig09]. A well-known example of workflows are *SAP Business Workflows*, which enable realisation of business processes in SAP enterprise systems including association of process steps with participating roles and persons. This reaches from simple approval processes to complex sales or marketing processes [Gat09]. Moreover, workflows are typically used to enable large-scale scientific operations in grid computing environments [DC08, BC09].

Note that in some related work there is no terminological differentiation between the business and the technical level of business processes. Yet, in order to ensure clear separation of business processes (business level) and their detailed modelling enabling IT-supported execution (technical level), this thesis sticks to the term workflow.

Workflow Model. Workflows – or being more precise workflow models – consists of *activities* that are interconnected by means of *transitions*. The WfMC defines the term activity as "a description of a piece of work that forms one logical step within a process" [Coa99, p. 13]. Activities are typically assigned the executing resource, dependencies on other activities, as well as resources needed for activity execution. This may include executable code, user interface options and media required for user interaction, or additional information consumed by the user during activity processing [SSZA$^+$11].

Transitions interconnecting activities may reflect clear predecessor / successor relations resulting in purely sequential structures. In addition, [Coa99] defines *AND-split* as well as *AND-join* transitions, which enable modelling of parallel structures as well as of join points for converging parallel activity sequences, respectively. Furthermore, *OR-split* and *OR-join* transitions support modelling of workflows with alternative structures. In order to enable workflow engines deciding on which branch to follow during runtime, OR-splits are typically conditioned by means of annotations capturing logical expressions.

Moreover, this thesis takes into consideration the notion of *XOR-split* and *XOR-join* transitions. In fact, OR transitions may lead to multiple of the resulting workflow branches being followed in parallel, while XOR-splits allow only a single of the workflow branches to be processed. A motivation of this differentiation with respect to the proposed workflow-based replication strategies is presented in Section 4.1.1.

Workflow Execution. According to [MSMP11], this thesis distinguishes two flows related to workflow execution. First, the *control flow* captures processing of the activity logic in the sequence defined by the workflow structure. In related work, this flow is sometimes also referred to as process flow. Second, the *data flow* covers content retrieval and placement operations needed for staging-in activity-related content needs and staging-out results of the actual activity processing [DC08].[14] A discussion of different degrees of coupling and synchronisation between the control flow and the data flow of workflows is presented in Section 4.1.1.

Summarising the above paragraphs, workflow management includes (i) the specification and design of business processes, (ii) the modelling of workflows as basis for IT-supported process execution, as well as (iii) the actual workflow execution that is typically driven by process-aware information systems or dedicated workflow management systems [Rig09].

The terms being most relevant to the content placement strategies defined in Section 4 are summarised in the following paragraph.

- **Activity-related Content Needs:** Resources needed for processing an activity are referred to as activity-related content needs.
- **AND / OR / XOR Transitions:** For reasons of simplicity, AND- / OR- / XOR-splits are referred to as AND / OR / XOR transitions. The corresponding join points are not taken into consideration. This is reasoned by the scope of the proposed workflow-based replication strategies, which do not deal with potential synchronisation issues resulting from parallel activity sequences.
- **Alternative Paths.** Workflow branches resulting from OR / XOR transitions are referred to as alternative paths of a workflow.
- **Control Flow.** The control flow of a workflow captures processing of the activity logic in the sequence defined by the workflow structure.
- **Data Flow.** Given the objectives of the proposed workflow-based replication strategies, the data flow of a workflow covers the staging-in of activity-related content needs, only.

[14] The complete workflow-related content life cycle covering the phases workflow creation, planning, and execution is presented in [DC08].

2.2 Smart Products

2.2.1 Definition and Challenges

The era of ubiquitous computing envisages zillions of smart things (also known as smart items and smart objects) that interact with each other as well as various users across their life cycle. Such smart things range from things being equipped with smart labels (e.g., RFID or NFC tags) to resource-rich smart things with embedded storage, computing, and extended networking capabilities [MF10, Fle10]. Driven by advances in technology as well as research and standardisation efforts, the daily use of smart things is on the verge of becoming a commodity.

In general, smart things are referred to as physical entities with associated software components that are either deployed and operated on-board of these entities or by the environment surrounding the latter. Depending on the research community, smart things are characterised – amongst others – by being capable of adapting to situations and users [MV08] or by enabling real-world-aware business processes [Noc07]. This thesis follows the definition of *smart products* developed in the course of the EU-funded research project SmartProducts.[15]

As early as 2007, [Mü07b] stated two central properties to be addressed by smart products, *simplicity* and *openness*. In our today's world, everyday users tend to be challenged by the increasing complexity and diversity of technical products as well as the amount of associated information [MSH09]. *Simplicity* subsumes the need for enhancements in product-to-user interaction and aims at addressing the afore-stated issue by means of personalisation, multi-modality, and active user guidance. *Openness* captures the need for smart products to embed and operate in varying environments taking into consideration the entire product life cycle. These potential environments can neither be fixed nor anticipated at the design time of smart products. For this reason, smart products have to be able to self-organise as well as to cooperatively create their own "infrastructures" in a bottom-up approach (i.e., enhanced product-to-product communication). These requirements are addressed by the definition of smart products developed in the course of the SmartProducts project.

> *"A smart product is a hybrid entity, consisting of a physical object and a software object. It is capable of self-organised embedding into different environments over the*

[15] Note that the terms *environment*, *context*, and *life cycle* are also used in their definitions developed in the course of the SmartProducts project [SMVU11]. The definitions are summarised in the document's glossary presented in Annex B.

course of its life-cycle. It uses resources, such as storage, input or output capabilities and knowledge, which are built-in or provided by the environment, with the goal of natural and purposeful product-to-human interaction. At the same time, it provides built-in resources and knowledge in the environment for other smart products to use (product-to-product interaction).” [SMVU11, p.11]

According to the definition, smart products are tangible objects that possess enough resources for locally operating a minimal software component. Moreover, depending on their capabilities, smart products pursue or serve a well-defined purpose at a given point in time by means of associated knowledge and knowledge-related functionality. Yet, smart products are typically resource-constrained with respect to their communication and interaction, as well as their processing capabilities.

Due to their heterogeneous communication technologies and their potential mobility, smart products are subject to regular disconnections. Indeed, connectivity cannot be assumed in all environments smart products may be operating in across their life cycle. While connectivity is generally available in well-equipped manufacturing scenarios such as the smart aircraft manufacturing scenario presented in Section 2.2.3, domains such as mobile field services or logistics are typically characterised by occasional connectivity. Hence, *the core functionality of smart products must not rely on continuous connectivity to remote systems providing additional services and data* [MEA+11]. Furthermore, smart products typically possess limited storage capacity, only. Hence, given the connectivity restrictions described afore, *smart products can be assumed to not being able to store all content required and created during their life cycle on-board*. The resource limitation of smart products is further detailed by the hardware-based classification proposed by [Had12].

2.2.2 Concepts

In order to achieve simplicity and openness, a conceptual basis for realising smart products was developed in the SmartProducts project [SMVU11, NUD11]. Figure 2.2 illustrates an extract of these concepts. The two concepts *problem-solving knowledge* and *distributed and secure storage* highlighted in bold cover the key features of smart products in the scope of this thesis and are further detailed in Sections 2.2.2.1 and 2.2.2.2, respectively. A high-level architectural overview of the components reflecting the concepts depicted in Figure 2.2 is presented in [MSH09].

Embedding in Environments. To address resource limitations of individual smart products, smart products have to be able to self-organise and cooperate in arbitrary environments across their life cycle. This means, smart products need to utilise resources

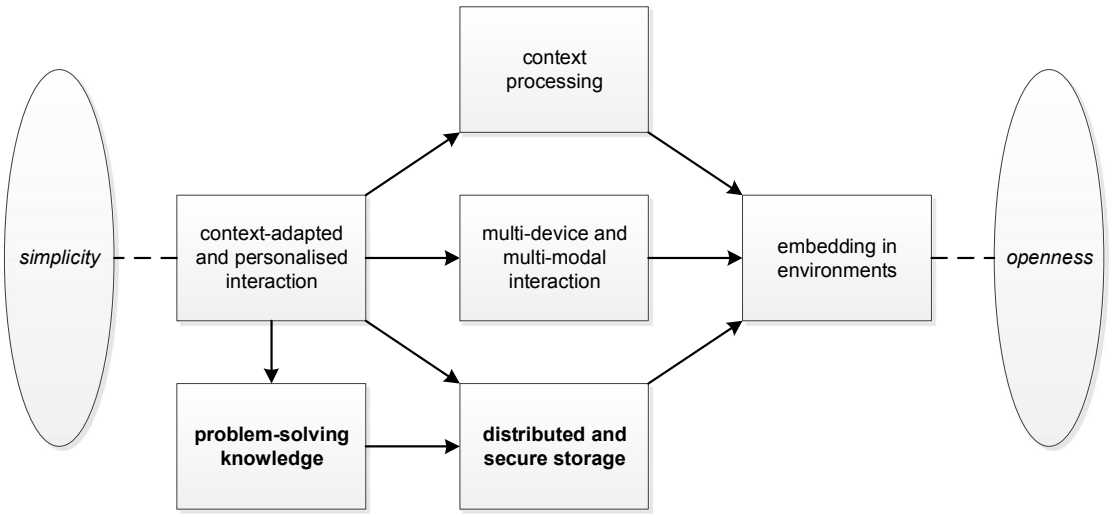


Figure 2.2: Overview of the Concepts of Smart Products (extract); dependencies are illustrated by directed relations between concepts (e.g., distributed and secure storage bases upon embedding in environments)

in the environment (e.g., sensor data, actuators for task automation, interaction devices) and make available their own resources (e.g., shared storage capacity) to facilitate operation of other smart products belonging to the same environment. This capability, subsumed by the concept *embedding in environments*, is central for achieving openness and provides the foundation for all other concepts of smart products.

Context-adapted and Personalised Interaction. Simplicity is addressed by the concept *context-adapted and personalised interaction*. This concept encompasses adaptation of user interaction according to user preferences and context, as well as user guidance that is enabled by smart products having information about pending tasks as well as knowledge required for task accomplishment. Hence, context-adapted and personalised interaction capabilities of smart products require (i) cooperative *context processing*, (ii) *distributed and secure storage* in order to retrieve user interface options given user preferences and context, (iii) *multi-device and multi-modal interaction* for rendering user interface options either locally or on suitable interaction devices in the environment, as well as (iv) *problem-solving knowledge* for (automated) task accomplishment.

Problem-solving Knowledge. Problem-solving knowledge linked with smart products consists of two types of knowledge. On the one hand, *procedural knowledge* is used to model well-structured procedures, which are often associated with today's industrial products and processes. Examples include work orders in the aircraft manufacturing domain and vehicle service and maintenance operations. The SmartProducts project recommends procedural knowledge to be modelled by means of workflows [SMVU11]. On

the other hand, *reasoning-based knowledge* provides the flexibility needed for cooperative problem solving in dynamically orchestrated smart product systems that do not allow for upfront process definition. Reasoning-based knowledge consists of tools and methods for dynamic process composition, task assignment, and task accomplishment [NdGV11]. Together, procedural knowledge and reasoning-based knowledge enable smart products to autonomously and cooperatively accomplish their tasks.

Distributed and Secure Storage. Finally, given the limited storage capacity of individual smart products as well as their potential intermittent connectivity, the concept *distributed and secure storage* enables distributed content storage and retrieval in varying environments while ensuring content-related security and privacy. Examples for the content handled by this concept include user models shared and maintained by multiple smart products as well as domain knowledge covering information about both the smart products domain and specific application domains. Moreover, problem-solving knowledge as well as user interface options and related media that might be – for example – used during workflow execution are maintained by the distributed and secure storage concept of smart products.

2.2.2.1 Workflow-based Operations

Workflows have been widely studied and proven as efficient means for enabling cooperative and automated task execution in mobile and smart environments, as well as for supporting active user guidance via instructional product-to-user interaction [KZL07, WNL08, Mey13]. This in particular applies to smart products, which pursue inherent and well-defined purposes. As of today, problem solving knowledge for well-structured procedures is typically described in the form of *imperative workflow models* [HUV10, HSU11, Sch11, SHS12] using workflow modelling languages such as XPDL for Ubiquitous Semantic Environments (XPDL4USE) that enable incorporation of semantic annotations [SSZA+11, WUS10].[16] Assuming the availability of suitable ontologies, the actual set of smart products participating in the execution of distributed workflows can be determined dynamically at runtime. This enables adaptation of workflows according to the composition of smart product systems [HSU11, Sch11].

The amount of workflow-based operations of smart products typically depends on the application domain. Workflow-dominated smart product applications can for example be found in the manufacturing domain. In the scenario *smart aircraft manufacturing* presented in Section 2.2.3, all assembly operations are workflow-guided and most of the participating smart products have a clear, singular purpose. This also leads to workflows being executed repeatedly. A smart torque wrench being shared by multiple workers can be taken as an example.

[16] An exemplary realisation of a workflow engine enabling distributed XPDL4USE workflow operations for smart products is presented in [DMG+11, MSB11].

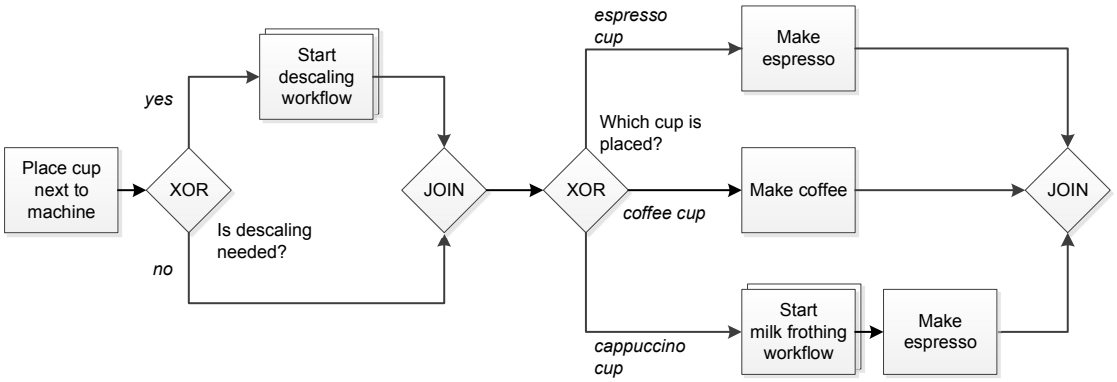


Figure 2.3: Exemplary Workflow of a Smart Coffee Maker

An exemplary workflow developed in the course of the SmartProducts project is illustrated in Figure 2.3 [SSZA+11].[17] This workflow is associated with a smart coffee maker and consists of two XOR transitions. The first XOR transition checks whether the machine is required to be descaled and – if so – starts a corresponding sub-workflow. Depending on the capabilities of the machine, this workflow could also consist of alternative paths, which would result in nested XOR transitions.[18] The second XOR transition of the workflow is conditioned by the cup placed by the user. Depending on the cup, the workflow branches into three alternative paths. In case of a cappuccino cup being placed by the user, there is a sub-workflow for frothing milk. Again, this workflow could consist of alternative paths resulting in nested XOR transitions.

Obviously, the structural complexity of workflows results from – potentially nested – OR / XOR transitions as well as synchronisation issues in case of parallel work streams. Workflow branches are typically context-dependent or affected by the outcome of the activity preceding the OR / XOR transition (e.g., measurements or explicit user selection [SSV+12]). While the above-shown workflow is used for initial demonstration purposes, one can expect much more branched workflow models in future scenarios, in particular in complex manufacturing and maintenance scenarios.[19]

[17] For reasons of simplicity, semantic annotations and activity-related content needs are not shown. Examples for the latter are presented in Section 2.1.5.

[18] An exemplary workflow for descaling the smart coffee maker presented in [TKL11] is proposed by [SSZA+11].

[19] Additional workflows associated with the use cases developed in the SmartProducts project are presented in [SSV+12]. Also, the cocktail companion demonstrator proposed by [TKL10] fully relies on workflow-based operations.

2.2.2.2 Distributed Storage System

Content management in the ubiquitous computing domain has been studied intensively in the last decade. As stated by [CFZ01], "pervasive computing brings challenges in all aspects of data management" [CFZ01, p. 139], which "plays a crucial role" [CFZ01, p. 139] for achieving the vision and objectives of the former.

As stated in Section 2.2.1, smart products are typically not able to locally store all content required and created across their life cycle neither can they connect to remote systems at all times for storing and / or retrieving information. Moreover, content needs of smart products may vary depending on the location and context of the products, as well as the actual users working with the latter [CFZ01]. Also, the potential operation of smart products in disconnected environments and the resulting network partitioning may require content to be available locally either on-board of the smart product or on resources of the environment the product is / will be used in [PJFY04].

Because of the dynamic composition of smart environments with smart products playing the role of both producers and consumers of information as well as the overall scalability challenges of ubiquitous computing, there is a "need for intelligent data staging and pre-staging, so that data can be placed close to where the users will be when they need it" [Fra01, p. 4]. Additional information on challenges of and requirements for content management strategies in the ubiquitous computing domain is presented in [CFZ01, Fra01, PJFY04, Bot09, Com09].

In the last years, the World Wide Web Consortium (W3C) Object Memory Modeling incubator group worked on a specification of an object memory format that enables modelling and storage of life cycle data of smart things.[20] Also, the group proposed a terminology and a classification scheme for storage frameworks targeting Digital Object Memories (DOM) associated with smart things [MKH11]. An analysis of envisaged and partially prototyped DOM application scenarios revealed a broad set of applied content management solutions and resulted in the four classes of storage frameworks presented in Figure 2.4.

First, in the approach depicted in Figure 2.4(a), each smart thing possesses an identifier only and all content of its DOM is centrally stored in a remote system [BLMHS+13]. Second, the class illustrated in Figure 2.4(b) represents distributed storage systems such as the Object Memory Service [Sch07] that rely on a central index store linking content of the DOM that is stored in multiple remote systems [SMK+10]. Third, the class depicted in Figure 2.4(c) extends the former by supporting distribution of the DOM between remote systems and the smart thing the DOM is associated with [KGB+11] Last, Figure 2.4(d) shows a fully distributed solution that makes use of local resources in the environment the smart thing is operating in.

[20] http://www.w3.org/2005/Incubator/omm/

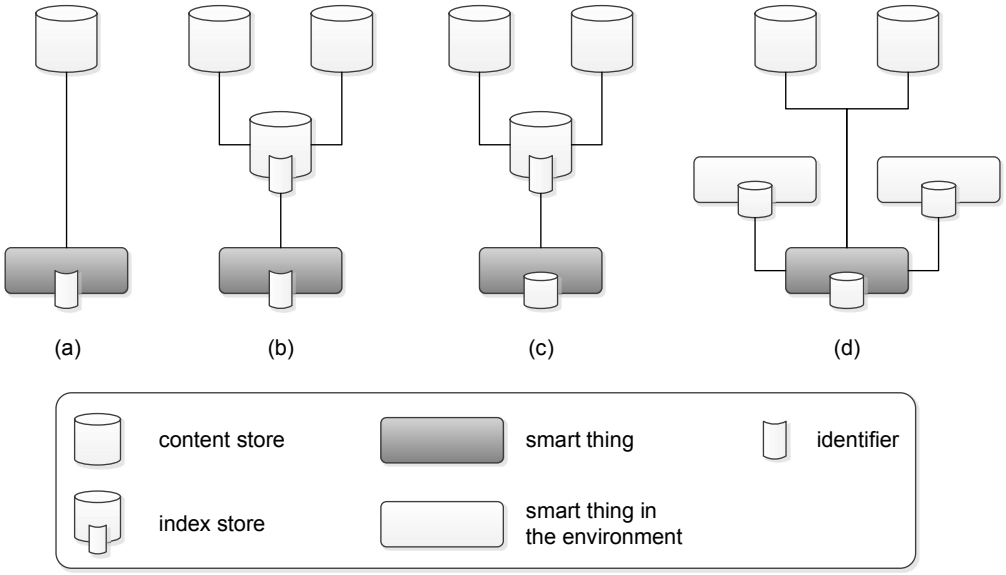


Figure 2.4: Storage Frameworks for Digital Object Memories associated with Smart Things

Given the challenges of the smart products domain – in particular the non-guaranteed connectivity to remote systems – a fully distributed storage system is the only applicable approach for a generic content management solution (see Figure 2.4(d)). A detailed description of a distributed and secure storage concept of smart products (see Figure 2.2) that aims at enhancing content availability, persistence, and query efficiency properties while preserving security and privacy of distributed content is presented in [MEA+11, Bec11].[21] The author of this thesis contributed to the definition and description of this functionality, which presents the basis for the proposed content placement strategies.

This distributed and secure storage concept of smart products includes on- and off-board storage functionality that transparently places content either locally or remotely on resources in the environment or remote system. Moreover, optimistic replication strategies combined with replacement mechanisms as well as configurable quorum-like consistency maintenance policies are proposed. Finally, the search functionality of the distributed and secure storage concept enables location and retrieval of content distributed within and across environments or provided by remote systems (if connectivity can be established) by means of exact-match queries, keyword search based on a controlled vocabulary of content metadata, as well as complex queries.

[21] Note that even though security and privacy features of the distributed storage system are of high importance in the envisaged application scenarios, they are out of scope of this thesis.

Content Demand Behaviour of Smart Products. The content demand behaviour of smart products adopts the locality properties temporal locality and geographical locality explained in Section 2.1.2. The purpose-driven nature of smart products is a clear indicator of *temporal locality*. This in particular applies to smart products that repeatedly process a small set of associated workflows. A smart coffee maker or a smart torque wrench used by service technicians represent examples. Note that even though such smart products typically access a certain content set regularly, there might be content requests that vary depending on the environment the products belong to as well as the users operating the latter (think of varying user interface options and media depending on the available interaction devices).

Geographic locality properties can be assumed in environments consisting of cooperating smart products. Manufacturing scenarios in which multiple workers jointly process complex workflows can be taken as an example. Even though each worker might focus on different activities within the workflow they typically all require central information about the component they are working on (see Section 2.2.3 for an exemplary scenario).

2.2.3 Use Case: Smart Aircraft Manufacturing

Today's aircraft manufacturing processes are characterised by paper-based process descriptions that have to be carried by blue-collar workers as well as manual tracking of assembly results. Obviously, the integration of smart products such as smart torque wrenches has great potential for enhancing efficiency of aircraft manufacturing processes. This includes automation of process steps such as tool configuration, maintenance of results of single process steps, as well as accumulation and aggregation of assembly results in order to obtain (real-time) information about the manufacturing state of aircrafts.

Figure 2.5 provides an overview of a future *smart aircraft manufacturing* scenario as envisaged by EADS Innovation Works that was developed jointly in the course of the SmartProducts project [HG10]. The illustration shows the different kinds of smart products used in the scenario as well as their organisational interrelation.

In this scenario, instance-level aircraft manufacturing process descriptions are distributed from the aircraft manufacturer to the different plants, in which they are further dispatched to the corresponding smart aircraft components and / or workstations at which manufacturing is accomplished. Plants consist of several workstations with multiple blue-collar workers per workstation. Each blue-collar worker carries a nomadic device, which is used to display instructions of manufacturing processes as well as to annotate corresponding results. In addition, each blue-collar worker makes use of smart tools (e.g., smart torque wrench, smart drill) that are automatically configured based on extended manufacturing process descriptions, i.e., workflow models, and capable of delivering results (e.g., torque and angle) to nomadic devices. Even further, while most

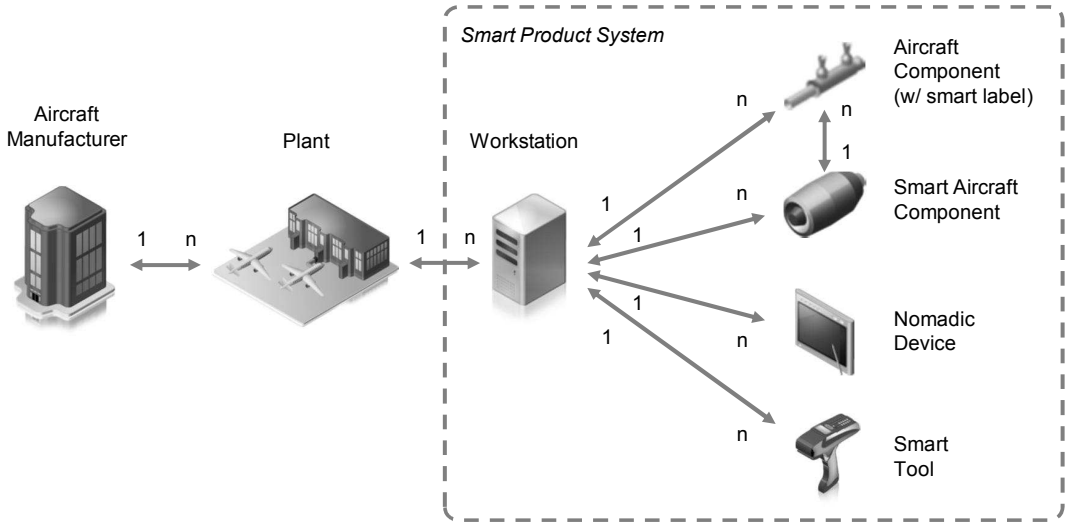


Figure 2.5: Smart Aircraft Manufacturing

components such as bolts are merely identifiable via smart labels, certain (compound) aircraft components also play the role of smart products. This way, they are able to maintain digital representations of their manufacturing state as well as manufacturing processes that are to be accomplished (see [KHB+11]). Eventually, the component-level information maintained by smart aircraft components and workstations is delivered to and aggregated by the aircraft manufacturer in order to realise digital representations of entire aircrafts as being manufactured [HG10].

The smart products of the smart aircraft manufacturing scenario all face the challenge of limited storage capacity and incomplete network coverage in plants [MEA+11]. This points out the need of a distributed storage service with dedicated content placement strategies. This service needs to maintain a high-level of user-perceived performance when accessing required information (i.e., query efficiency). Moreover, the service needs to aim at enhancing content availability and persistence in order to ensure complete digital representations of aircrafts as manufactured that can be used in subsequent phases of the product life cycle.

The relevance of the scenario can be illustrated by SAP's solution offering in the domain of mobile enterprise asset management [AG12]. In fact, this solution can be classified as intermediate step towards smart enterprise asset management scenarios, in which mobile devices directly interact with the assets as well as the tools they are operating with.

2.3 Roles of Smart Products

In order to ease the description of the proposed content placement strategies, this section defines the roles smart products may play in a distributed storage system as well as during the exchange of related messages. The explanation of these roles is exemplified by the smart product system depicted in Figure 2.6.

Smart Product System. A smart product system encompasses all smart products that occasionally or continuously participate in a smart product application scenario. A smart product system does not have to be limited to a single environment but may cover multiple environments independent of their interconnection. Non-smart products used within a smart product application scenario as well as the infrastructure enabling communication among smart products are classified as indirect elements of a smart product systems.

Furthermore, a smart product system may include remote systems, which are assumed to provide persistent and virtually infinite storage resources as well as broadband communication access. In order to capture not only enterprise systems in the traditional sense (e.g., SAP ERP), but moreover cloud services (e.g., Amazon S3) that feature the above-stated characteristics, such remote systems are subsumed under the umbrella term *cloud*.[22] For example, the computational infrastructure of the aircraft manufacturer's back office depicted in Figure 2.5 is used to persistently store digital representations of aircrafts as being manufactured. Consequently, it plays the role of a *cloud* entity given the above explanation. In related work, this class of entities is typically denoted as "back-end system" [Noc07, SMVU11].[23]

Figure 2.6 shows an exemplary smart product system consisting of seven smart products $\{SP1, .., SP7\}$ of which products $\{SP5, SP6\}$ and $\{SP4\}$ store content Ci and related replicas Ri, respectively. The relation between content and replicas is denoted by the numbering i. Thus, the object $R1$ is a replica of content $C1$, while there are no replicas associated with content $C2$. Moreover, the illustration covers storage – *PUT* – and retrieval – *GET* – requests for content Ci labelled as P_{Ci} and G_{Ci}, respectively. Note that in both cases the smart product associated with a request is the one triggering the latter. Hence, the content retrieval request G_{C1} is dispatched by the smart product $SP1$ and can be served by either of the products $\{SP4, SP6\}$. The content storage request P_{C2} is triggered by the smart product $SP2$. For reasons of simplicity, it is assumed that the storage request P_{C2} is served by the smart product $SP5$.

[22] A definition of the term *cloud computing* in its general sense including the related service models is provided by [MG12, fSidI13].

[23] This abstraction does not limit remote systems to a single component. Remote systems could be organised in structures that enhance content distribution in large-scale smart product systems (e.g., following organisational or geographic structures). Yet, structuring of remote systems is not detailed in the scope of this thesis.

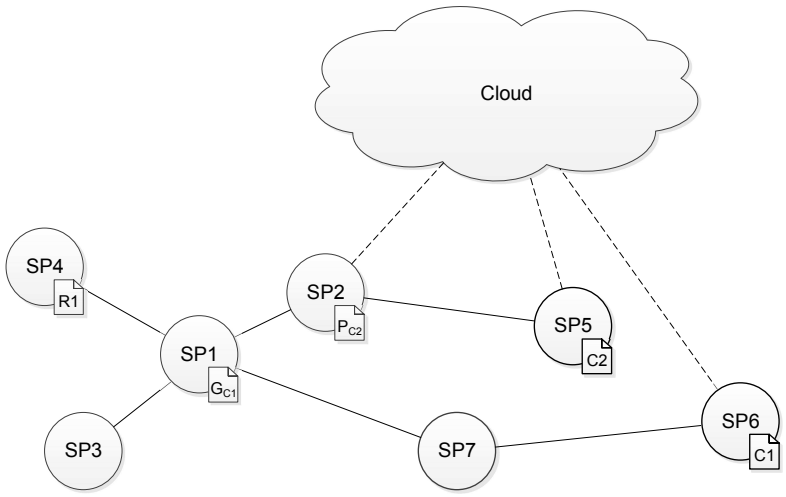


Figure 2.6: This figure illustrates an exemplary smart product system with seven smart products $\{SP1, .., SP7\}$ and a cloud component. The elements Ci and Ri represent content and related replicas, respectively. Moreover, the elements G_{Ci} and P_{Ci} illustrate retrieval (*GET*) and storage (*PUT*) requests for content Ci. The smart product associated with such a request is the one triggering the latter.

The solid edges interconnecting smart products as well as the dashed edges between smart products $\{SP2, SP5, SP6\}$ and the cloud represent arbitrary, logical communication links and are not meant to reflect realistic topologies by any means. This logical topology shows a non-fully-meshed structure, which might imply multi-hop communication. This is reasonable, because it cannot be assumed for all use cases that smart products maintain direct communication links to all other smart products of the specific system. For example, such an organisation might occur in case of smart products being equipped with different communication technologies or in case of smart products being organised in overlay network structures.

2.3.1 Distributed Storage Roles

From the perspective of distributed storage systems, this thesis distinguishes four main roles smart products may play. Since any of these roles can be taken on either by smart products or by an entity of the class cloud, the generic term *node* is used in the following definitions.

Initiator. A node that dispatches a *PUT* request in order to initially inject content into the distributed storage system is referred to as *initiator*. In the example presented in Figure 2.6, the node $SP2$ plays the role of an initiator for the content object $C2$.

Provider. A node storing content of any type on-board is called a *provider* of this content. A provider contributes a set of content objects to the distributed storage system and is capable of serving corresponding *GET* requests. In the example presented in Figure 2.6, this role is taken on by the nodes $\{SP4, SP5, SP6\}$.

Primary provider. The role *primary provider* is a special case of the provider role. While the latter captures all nodes that store content of any type, the primary provider is only played by nodes that initially store a content object after it was injected into the distributed storage system by an initiator. Hence, the role primary provider covers content objects only; replicas always result from subsequent operations. Based on the assumption that the two content objects $C1$ and $C2$ were not replaced, the role of a primary provider is exemplified by the two nodes $SP6$ and $SP5$, respectively, in the smart product system presented in Figure 2.6.

Requester. A *requester* is a node that queries content by means of a *GET* request. It can also play the role of a provider in case it stores the requested content on-board, i.e., in case its *GET* request is served locally. In the example presented in Figure 2.6, the node $SP1$ plays the role of a requester for the content object $C1$.

Moreover, the thesis defines two classes of nodes from the perspective of a smart product that can be used by content placement strategies to enhance content access.

Neighbour. Nodes that reside in the spatial proximity of a given node are referred to as the node's *neighbours*. Neighbours are assumed to enable low-latency access to the content they provide. Nodes maintain neighbours in neighbour sets that are sorted in descending order of their proximity given dedicated proximity metrics.

Content repository. Nodes that enable reliable content location and retrieval in the distributed storage system are referred to as *content repositories*. Again, nodes maintain content repositories in content repository sets that are sorted in descending order of specific metrics.

In the example presented in Figure 2.6, the nodes $\{SP2, SP3, SP4\}$ could represent neighbours of the node $SP1$, while nodes $\{SP5, SP6\}$ could play the role of content repositories of the latter. The implementation presented in Section 5.3 includes a distributed storage service that operates on top of the structured P2P overlay network Pastry [RD01a]. Pastry supports both content repositories and neighbours by means of its state tables *leaf set* and *neighbour set*, respectively. These state tables are used to enable proximity-aware routing and support the distributed storage service to enhance content locality.

2.3.2 Message Delivery Roles

Direct & Indirect Requests. In distributed storage systems, nodes communicate by exchanging messages in order to organise content placement as well as to cooperatively serve *PUT* and *GET* requests. In general, one can distinguish two types of requests. Requests that originate from applications and / or services that are deployed on a node that operates an instance of the distributed storage service for request handling are referred to as *direct requests* from the perspective of this node. If direct requests cannot be served locally, then the instance of the distributed storage service takes care of disseminating them in the distributed storage system following specific, pre-define rules for message delivery. Such forwarded requests are named *indirect requests* regardless of whether they can be served by the receiving node. Indirect requests are always based on direct requests.

For example, the *GET* request G_{C1} is classified as a direct request from the perspective of the requester $SP1$. Depending on the strategy applied by the distributed storage service, the request could be served by either of the nodes $\{SP4, SP6\}$, which are both providers of the requested content $C1$. Both providers would classify G_{C1} as an indirect request. This also applies to the *PUT* request P_{C2}, which is classified as a direct request by the initiator $SP2$ and as an indirect request by the node $SP5$ serving the request.

Moreover, the thesis defines the following message delivery roles that might be taken on by any of the nodes participating in the distributed storage system. While these roles are not limited to any specific request, a consistent example is used for illustration purposes based on the smart product systems presented in Figure 2.6. The node $SP1$ requests content $C1$. Since $SP1$ is not a provider of the requested content, it sends a *GET* request message for content $C1$. It is assumed that this message is delivered to node $SP7$, which forwards the message to node $SP6$. $SP6$ is a provider of the requested content and sends a *GET* response message back to the requester $SP1$. This example is illustrated in Figure 2.7.

Source. A node that initially creates and dispatches a message is referred to as the *source* of the message. In the above example, the nodes $SP1$ and $SP6$ play the role of the source of the *GET* request and *GET* response message, respectively.

Destination. A node targeted by a message is referred to as *destination*. This node can be specified either explicitly (e.g., by means of the node's IP address) or implicitly. For example, structured P2P overlay networks typically use node identifiers for implicit specification of the message destination (see Section 2.1.4). Note that if the destination cannot serve the request encapsulated in the message, it may forward the message and – in retrospect – take over the role of an intermediate as described below. In the above example, the nodes $SP6$ and $SP1$ play the role of the destination of the *GET* request and *GET* response message, respectively.

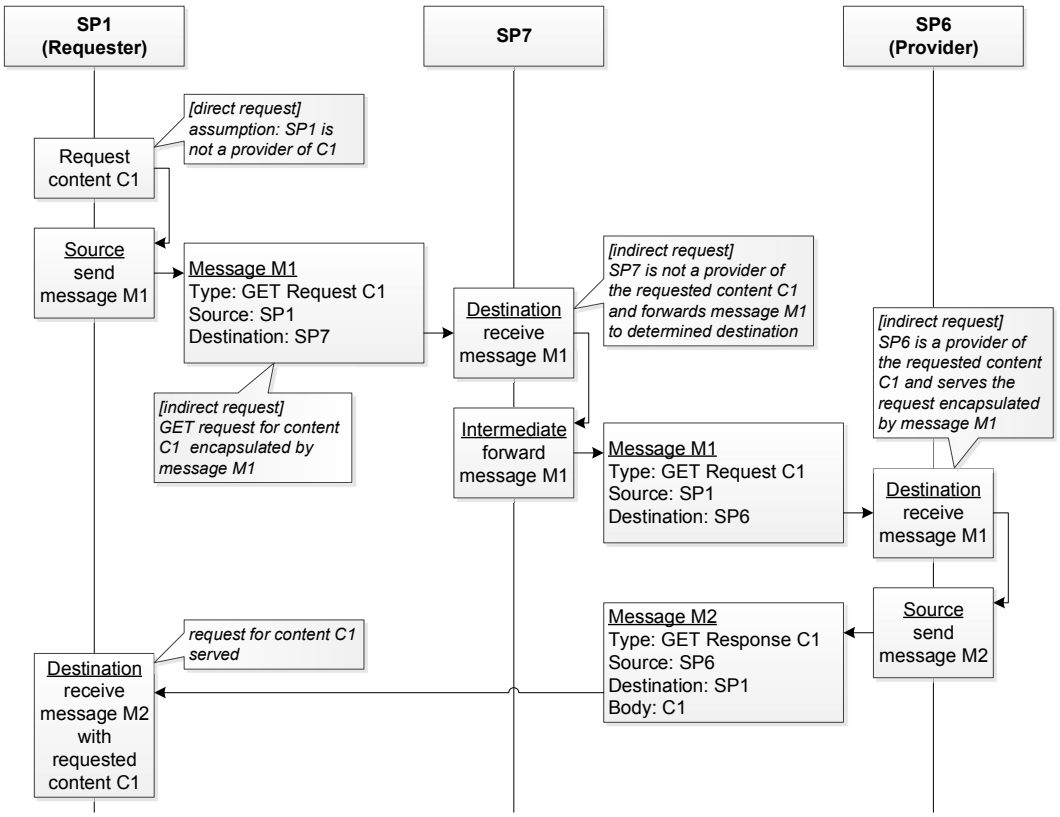


Figure 2.7: Simplified Process Flow of a *GET* Request

Intermediate. A node on the delivery path from source to destination is referred to as *intermediate*. An intermediate receives messages either from source or other intermediates and forwards messages to the destination or other intermediates. In the above example, node $SP7$ plays the role of an intermediate with respect to the *GET* request message.

Message Delivery Types. Depending on the capabilities of the distributed storage service as well as of the underlying communication middleware, three kinds of message delivery are defined. Message delivery from source to destination by means of direct addressing (e.g., based on a node's IP address) is referred to as *message sending*. *Message forwarding* is a special case of message sending. It is used if a destination has to forward a message to another node, in case it cannot serve the request locally. In short, message forwarding denotes sending of a message from a node other than the message source. Also, messages might be *routed* to an indirectly specified destination. Plaxton's prefix matching approach [PRR99] for structured P2P overlay networks can be taken as an example.

2.4 Requirements for Content Placement Strategies in Smart Product Systems

In the course of the SmartProducts project, requirements for a distributed storage system for smart products with the features summarised in Section 2.2.2.2 were investigated based on a set of envisaged application scenarios [GKM+11, MEA+11]. Based in this analysis, this section lists and discusses the main requirements to be addressed by content placement strategies in order to achieve the goals stated in Section 1.2 given the challenges of smart products (see Section 2.2.1). The requirements for content placement strategies are clustered into four groups: *locality properties*, *knowledge of smart products*, *configuration and application control*, as well as *resource limitations and scalability*.

Locality Properties. The content demand behaviour of smart products is discussed in Section 2.2.2.2 following the locality properties of access patterns defined by [RF01]. The section concludes that the purpose-driven nature of smart products as well as their cooperative behaviour in dynamically-composed environments typically results in access patterns with temporal locality properties and geographic locality properties, respectively. On the one hand, knowledge of temporal locality properties can be inferred from past content requests by individual smart products. This knowledge is referred to as *local knowledge*. Geographic locality properties, on the other hand, can only be learned and leveraged by smart products exchanging knowledge about content interest on a regular basis. Consequently, this knowledge is named *cooperative knowledge*.

The combined knowledge of content demand behaviour should be leveraged by content placement strategies to implicitly predict and gain insights into upcoming demand (see Section 2.1.2), and adapt content organisation, accordingly. This need is reflected by the requirements $R1$ and $R2$ formulated below.

Requirement R1: Utilise Temporal Locality Properties
The content placement strategy shall utilise local knowledge about temporal locality properties of content access patterns of smart products.

Requirement R2: Utilise Geographic Locality Properties
The content placement strategy shall utilise cooperative knowledge about geographic locality properties of content access patterns of smart products belonging to the same environment.

Knowledge of Smart Products. As stated in Section 2.2.2, smart products leverage associated knowledge and knowledge-related functionality in order to achieve simplicity and openness. Amongst others, this may encompass context information, knowledge of learned mobility behaviour of smart products combined with location-dependent content interests, as well as problem-solving knowledge. This knowledge could be made use of for predicting upcoming content needs and enhancing number and placement of replicas.

While all knowledge could be utilised by content placement strategies, this thesis focuses on procedural problem-solving knowledge (see Section 2.2.2), which is one of the key features of smart products. For example, the application scenario presented in Section 2.2.3 fully relies on workflows that represent procedural problem-solving knowledge. Moreover, as stated by [BWYH06], "a process model based prediction of information retrieval / dissemination can be acquired and used for better efficiency and effectiveness." [BWYH06, p. 6].

The use of procedural problem-solving knowledge for optimising placement of content needed for task accomplishment can directly affect user-perceived performance of instructional product-to-user interaction. The pre-replication of user interface options and related media taking into account activities of a workflow as well as interaction devices and context of the environment in which the workflow is executed can be taken as an example. Hence, the utilisation of procedural problem-solving knowledge by content placement strategies contributes to the achievement of simplicity.

The requirements *R*3 and *R*4 summarise the concept of utilising problem-solving knowledge associated with smart products and enhancing user-perceived performance of guided product-to-user interaction, respectively.

Requirement R3: Utilise Knowledge of Smart Products
The content placement strategy shall utilise procedural problem-solving knowledge of smart products to actively place content where it is expected to be accessed.

Requirement R4: Enhance Performance of Guided User Interaction
The content placement strategy shall foster simplicity properties of smart products use by enhancing performance of guided product-to-user interaction.

Configuration and Application Control. Content placement strategies for smart products should not be tailored to the specifics of individual application scenarios. Instead, they should provide generic means for enhancing number and placement of replicas. For this reason, the strategies have to be configurable to enable system administrators to adapt them with respect to potentially varying characteristics of different smart product systems (see requirement *R*5).

This configuration should be complemented by simple means that allow application developers to (permanently) affect replica organisation. For example, similar to a service contract, applications should be able to specify that a particular content should be highly available or should be accessible with low latency. As opposed to an application-controlled content placement strategy, this results in a semi-transparent content organisation that goes beyond the traditional access-pattern-based concepts captured by requirements *R*1 and *R*2 (see requirement *R*6).

Requirement R5: Support Configuration
The content placement strategy shall be configurable with respect to the varying characteristics of different smart product systems.

Requirement R6: Support Application-affected Placement
The content placement strategy shall provide smart product applications with simple means for affecting number and placement of replicas.

Resource Limitations and Scalability. Finally, the limited storage capacity of smart products stated in Section 2.2.1 as well as the overall scalability challenges (in terms of message and computation complexity [Mü07a]) of the ubiquitous computing domain should be taken into account by content placement strategies (see Section 2.2.2.2). As stated by [CFZ01], there is a need for an "intelligent use of scarce resources to enhance data access" [CFZ01, p. 8]. For example, abstracting from correlated consistency issues, high replication degrees typically improve content availability, persistence, and query efficiency. Yet, this approach does not appear to be feasible given the resource limitations of smart products. This need is reflected by the requirements *R*7 and *R*8 formulated below.

Requirement R7: Consider Resource Limitations
The content placement strategy shall take into account resource limitations of smart products.

Requirement R8: Support Scalability
The content placement strategy shall be scalable given the scalability challenges of the ubiquitous computing domain.

2.5 Chapter Summary

This chapter provided an overview of concepts the novel content placement strategies are based upon. This includes distributed storage systems with a focus on content placement strategies covering both content replication and content replacement. Moreover, the content properties availability, persistence, and query efficiency were explained. The view on distributed storage systems was complemented by an overview of P2P overlay networks often used in smart product systems as well as an explanation of workflows and workflow-based operations leveraged by the proposed content placement strategies.

Thereafter, the domain of smart products was introduced covering a definition of smart products, an overview of their challenges and concepts, as well as the presentation of an application scenario. Again, special attention was given to workflow-based operations of smart products as well as related distributed storage concepts, which were described by adapting the afore-summarised concepts to the smart products domain.

The chapter moreover defined a terminology covering the roles smart products may play in a distributed storage system as well as during the exchange of related messages.

Finally, the chapter listed and explained requirements to be addressed by content placement strategies to achieve the goals stated in Section 1.2 given the challenges of smart products. An overview of these requirements is presented in Table 2.1.

Requirement	Description
R1	utilise local knowledge about temporal locality properties of content access patterns
R2	utilise cooperative knowledge about geographic locality properties of content access patterns
R3	utilise procedural problem-solving knowledge of smart products for active content placement
R4	enhance performance of guided product-to-user interaction
R5	provide means for configuration
R6	provide smart product applications with simple means for affecting number and placement of replicas
R7	take into account resource limitations of smart products
R8	be scalable given the scalability challenges of ubiquitous computing

Table 2.1: Overview of the Requirements that shall be addressed by Content Placement Strategies for Smart Products

Chapter 3
State of the Art

This chapter describes state-of-the-art content placement strategies and assesses the degree to which they address the requirements defined in Section 2.4. According to the weighting of the contributions of this thesis, the chapter presents the results of a detailed study on content replication strategies as well as an overview of content replacement strategies. The study focuses on four domains: P2P storage systems [MPV06, ATS04], Content Distribution Networks (CDN) [KKM02], Grid computing [ASD12], and client / server-based storage systems.[24]

The remainder of this chapter is structured as follows. Section 3.1.1 gives an overview of replacement strategies and points out the benefit of integrated content placement solutions. Thereafter, Section 3.1.2 presents a novel, overarching classification scheme for replication strategies. This scheme is used for categorising and comparing state-of-the-art replication strategies of the four aforementioned domains. Each replication strategy is summarised and assessed against the defined requirements as well as its applicability in the domain of smart products (see Sections 3.1.3 to 3.1.6). Moreover, given the active characteristic of the content placement strategies developed in this thesis, an overview of existing content access predictors is given in Section 3.1.7. Finally, Section 3.2 summarises the extent to which the presented strategies achieve the defined requirements and identifies and evaluates the closest work. The chapter closes with a short summary in Section 3.3.

This chapter is based upon and partly reuses descriptions out of the following publications of the author of this thesis: [MEA+11, MSB11]. Note that paragraphs reused from these publications are not marked as direct quotes if they were written exclusively by the author of this thesis. Quotations within reused paragraphs are taken over and marked accordingly.

[24] While the above-cited works link to surveys in the respective domains, [CBPS10] presents a general view on replication strategies.

3.1 Replication Strategies

3.1.1 Notes on Replacement Strategies

According to the definition of content placement strategies presented in Section 2.1.2 as well as the replication life cycle proposed by [ASD12], replacement strategies complement replication concepts by means of replica removal and relocation processes. In general, replacement strategies assess each content stored on-board with pre-defined cost and benefit metrics and remove or relocate content objects featuring the lowest benefit / cost ratios. On the one hand, benefit metrics typically assume access patterns with temporal locality properties. Thus, they take into consideration *recency* (i.e., when was the content last accessed) and *frequency* (i.e., how many requests occurred) that are often weighted by the period in which content requests took place. On the other hand, content size as well as the "effort" of content retrieval are commonly employed as part of cost metrics. A classification scheme and a comprehensive survey on replacement strategies is presented in [Smi82, PB03].

While many of the below-described replication strategies apply common replacement strategies such as LRU or Least Frequently Used (LFU), integrated concepts that tailor replica removal / relocation to the creation and distribution of replicas typically lead to enhanced replica organisation. Examples for integrated concepts include Predictive Hierarchical Fast Spread (PHFS) presented in Section 3.1.5.4 as well as the strategy Most Frequently Requested (MFR), which is proven to be highly effective (see Section 3.1.3.7). Moreover, OceanStore (see Section 3.1.3.1), PAST (see Section 3.1.3.3), and the pre-replication strategy described in Section 3.1.6.4 define multiple replica classes and provide class-specific replacement policies.

While the common replacement strategies such as LRU or LFU are assumed to be known to the reader, the integrated replacement strategies are presented as part of the associated replication mechanisms to point out their interrelation.

3.1.2 Classification Scheme

Past work extensively analysed challenges to be addressed by replication strategies of different application domains and proposed diverse means for categorising and comparing the latter [KKM02, KK04, On04, ATS04, ASD12]. This section presents a comprehensive classification scheme that consolidates these approaches and captures the central design decisions of replication strategies. The design decisions (level 2, level 2+3 for

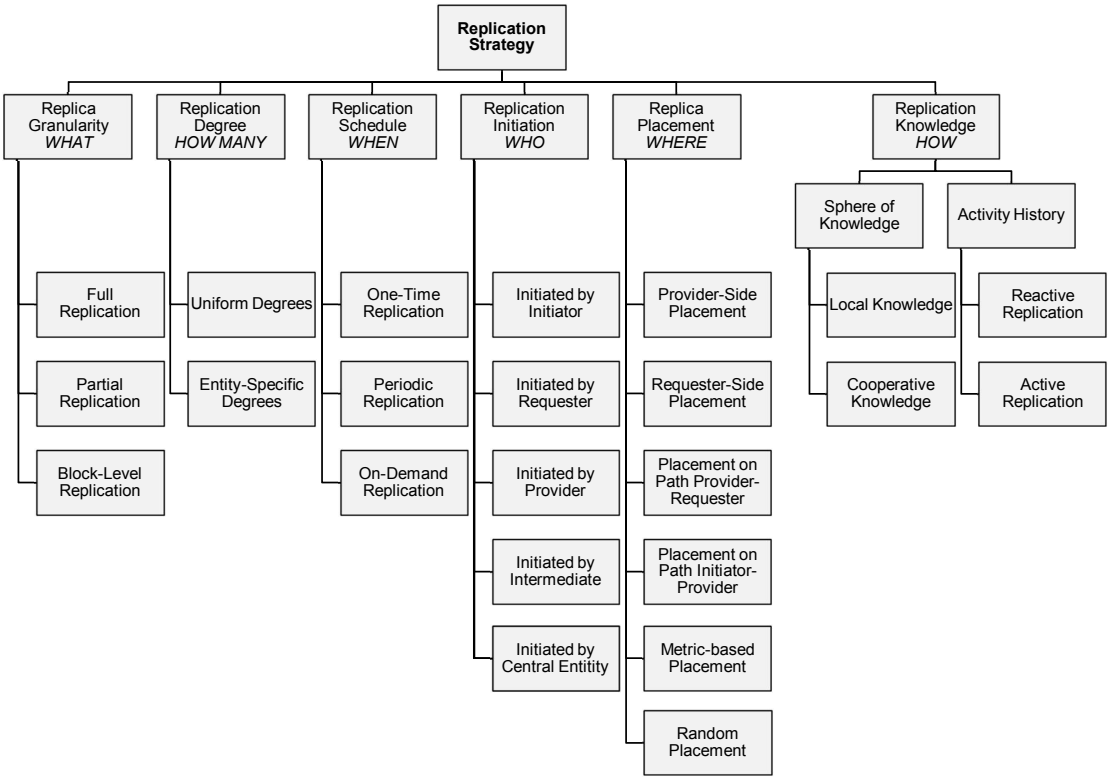


Figure 3.8: Design Decisions of Replication Strategies

replication knowledge) as well as alternative realisations (level 3, level 4 for replication knowledge) are illustrated in Figure 3.8 and explained in the following.

WHAT. The first design decision addresses the question of the candidate entity of replication, i.e., what to replicate. The proposed scheme differentiates three alternative *replica granularities*. Replication strategies with *full replication* maintain copies of the entire content set on different nodes (e.g., full database replication, also known as mirroring). *Partial replication* is more fine-grained and allows for content sets of different size to be used as candidate entity for replication. This reaches from multiple correlated content objects being replicated jointly to single content replication typically applied in P2P storage systems. Furthermore, *block-level replication* splits individual content objects into chunks, which are used as entity of replication. This approach is applied by erasure-code replication strategies.

HOW MANY. The design decision *replication degree* captures the number of replicas to create per entity of replication. The proposed scheme distinguishes between *uniform* replication degrees, i.e., the same number of replicas is created for all entities, and *entity-specific* degrees that support variation of the number of replicas for each entity of

replication. The latter supports more flexibility and facilitates, for example, replication strategies that adapt to changing access patterns.

WHEN. Another question to be approached is when the replication strategy is to be executed (*replication schedule*). Static approaches determine number and placement of replicas once (e.g., pre-configured or defined when content is stored initially). While the actual placement may change with system dynamics (think of structured P2P overlay networks that maintain the invariant of placing content on (surrogate) root nodes), static approaches do not actively adapt number and placement of replicas during runtime. For this reason, static replication scheduling is also referred to as *one-time replication*. In contrast, dynamic replication scheduling allows for re-assessment of number and placement of replicas during runtime. The classification scheme divides dynamic scheduling into *periodic* and *on-demand* approaches that enable execution of replication strategies in regular intervals and request-driven, respectively.

WHO. A further differentiator of replication strategies is captured by the node or role *initiating a replication process*. Note that even though multiple nodes may be involved in a single replication process, this design decision targets the one starting the process, only. According to the roles smart products may play in distributed storage systems (see Section 2.3), the scheme covers *initiator-*, *requester-*, *provider-*, and *intermediate-initiated replication*. Furthermore, replication processes can be triggered by *all nodes* (e.g., by means of a periodic assessment of number and placement of replicas made by all nodes) or by a *central entity* that manages the overall replication process.

WHERE. The question of *where to place replicas* is one of the main design decisions and may directly affect the extent to which the content properties defined in Section 2.1.3 are addressed by a replication strategy. Most often, strategies place replicas on-board or nearby the provider (i.e., *provider-side placement*) or requester (i.e., *requester-side placement*). While the former typically improves content availability given system dynamics, the latter targets improvement of query efficiency assuming access patterns with a certain level of temporal locality. Moreover, replicas may be placed on intermediates on the path from *provider to requester* or from *initiator to provider*. Finally, the selection of replica providers may follow specific *metrics* (e.g., selection of nodes with long uptime) or can be made *randomly* (e.g., for placing erasure-coded blocks on distinct nodes in order to improve content persistence).

HOW. Finally, the design decision *replication knowledge* encompasses the information taken into consideration for making replication decisions. The classification scheme divides this design decision further into the *sphere of knowledge* and the *activity history* utilised by the replication strategy. On the one hand, the sphere of knowledge can be limited to *local knowledge* of the node making a replication decision. On the other hand, such a node may exchange knowledge with other nodes in order to make replication decisions *cooperatively*. The difference between local and cooperative knowledge can be clarified with a simple example. Imagine a P2P storage system with nodes maintaining information about the surroundings as part of the applied routing mech-

anism. If this information is leveraged for content placement, it is referred to as local knowledge, because it is not gathered actively by the replication strategy. In contrast, if nodes in this system exchange information (e.g., about their content interests) as part of the replication procedure and this knowledge is utilised for making informed replication decisions, then this knowledge is classified as cooperative knowledge. In addition, note that depending on the overall system design both local and cooperative knowledge may represent global knowledge in case of a centralised system and a node exchanging knowledge with all other nodes for making replication decision, respectively. Hence, the sphere of knowledge denotes the number of nodes that actively exchange information required for making replication decisions.

The second sub-decision, *activity history*, distinguishes between reactive and active replication strategies. On the one hand, *reactive strategies* base their decisions on past observations, only. *Active replication strategies*, on the other hand, include mechanisms for estimating upcoming demand and system state for optimising content access.[25] Even though active strategies are known to greatly improve content availability and query efficiency, they may have to balance the risk of false replication.

Means for estimating future demand, i.e., recognising access patterns, include statistic demand forecasting [AMAM04], sequence prediction algorithms (see Section 3.1.7), explicit application hints [VAK+98, PGS93, YFLS11], implicit application hints by means of annotated workflow models [BWYH06, CDL+07] or user profiles [CGFZ03], and data mining concepts to infer semantic relationship between content [KWZ+00, JIR+09, KIS11]. A replication strategy that takes into consideration correlated content access patterns instead of purely relying on access frequency and recency is proposed by [TY14]. The presented simulation results point out the value of replica placement decisions being made based on content correlation, which can – for example – be inferred by means of data mining concepts or annotated workflow models.

Mechanisms for predicting upcoming system state encompass, amongst others, prediction of node failure behaviour to prevent content loss [MN05, DBEN07], prediction of node movement combined with known / learned location-dependent content interests [KKREM04], as well as mechanisms that leverage social relationship between nodes to organise content placement [BCP08].

In the subsequent sections, the proposed classification scheme is used to categorise related work. Moreover, in Section 4.5, the scheme is adapted to the novel content placement strategies in order to ease comparing the latter with related work.

[25] Note that for reasons of simplicity replication strategies that create replicas when content is initially stored are also classified as "reactive", because they do not use any prediction of upcoming demand. Indeed, these strategies could be classified with activity history "none". Yet, it is assumed that the proper selection of the number of replicas to create during initial storage is based on knowledge gained in past periods. Similarly, strategies that assume global knowledge (e.g., static approaches) are classified as active strategies.

3.1.3 Peer-to-Peer Storage Systems

This section summarises replication strategies used in P2P storage systems. A survey of P2P replication strategies is presented in [MPV06, ATS04]. Moreover, an overview of P2P overlay networks that provide the basis for P2P storage systems and related replication strategies is presented in Section 2.1.4.

3.1.3.1 OceanStore

OceanStore [KWZ+00, Gee02] is advocated by the authors as a highly scalable and available distributed storage system built on top of the structured P2P overlay network Tapestry [ZKJ01, ZHS+04]. To achieve high levels of content availability, persistence, and query efficiency, OceanStore applies a set of replication strategies.

Oceanstore Put. Content availability is addressed by the primary provider distributing replicas to a pre-defined number of nodes. Given the use of Tapestry, these nodes are determined by hashing the content identifier with known salt values.

OceanStore Deep Archival Storage. Moreover, in order to optimise content persistence given node churn, OceanStore applies an erasure-code replication strategy called *Deep Archival Storage*, which creates and distributes erasure-coded blocks for each committed content version. The strategy employs object-specific stretch factors and selects nodes to place erasure-coded blocks based on their reliability.

OceanStore Caching. OceanStore optimises query efficiency by placing so called *secondary replicas* on the requester side. Secondary replicas are soft-state objects that can be created and deleted as needed and complement *primary replicas* used to ensure consistency. Secondary replicas are placed on-board of the requester after receiving the requested content.

OceanStore TTL-based Replication. OceanStore enables requesters to assign Time To Live (TTL) properties to their requests. If this TTL expires prior to locating a provider of the requested content, the node that last processes the request tries to create a local replica to improve future access.

Oceanstore Introspective Replica Management. OceanStore employs a collaborative introspection approach to identify access patterns and dynamically optimise number and placement of replicas. This enables replica management, which includes the creation of additional replicas on the provider side in case of a provider tending to become overloaded as well as the removal of unused replicas in order to maintain high levels of replica utilisation.

With regard to the requirements for content placement strategies in smart product systems investigated in Section 2.4, the two first-explained strategies, i.e., OceanStore

Put and OceanStore Deep Archival Storage, address none of the requirements R1 to R6. The other three replication strategies employed by OceanStore all utilise temporal locality properties of access patterns (R1). While the introspective replica management approach does not use any type of problem solving knowledge, it is still classified as an active replication strategy. Hence, requirement R3 is rated as partly addressed. Similarly, although neither of the two request-oriented strategies, i.e., OceanStore Caching and OceanStore TTL-based Replication, improve product-to-user interaction, they both enhance access performance (R4 partly addressed). Finally, OceanStore is presented as a highly scalable distributed storage system that takes into account heterogeneity of storage node. Hence, both requirements R7 and R8 can be assumed to be met for all replication strategies of OceanStore.[26]

3.1.3.2 Freenet

Freenet is a distributed storage system that operates on a purely decentralised and loosely structured content location and routing substrate [CSWH01, ATS04]. Freenet makes use of two replication strategies and aims at improving content availability and query efficiency. Moreover, Freenet applies a LRU-based replacement (removal) strategy in order to account for the limited storage capacity of nodes and to enhance replica utilisation.

Freenet Insert. Freenet applies a two-phase store procedure with phase 1 capturing the determination and validation of a path from initiator to provider with an initiator-defined path length. In phase 2, the content to be stored in the network is sent along the path established in phase 1 and all nodes on the path try to create a local replica.

Freenet Request. Similar to the store procedure, content retrieval lets all nodes on the path from provider to requester create a local replica. Note that in contrast to other P2P-based distributed storage systems, Freenet does not apply direct delivery of responses from provider to requester but follows the route of the request message in reverse order.

The two replication strategies of Freenet can be classified as addressing requirement R1. Similarly, although Freenet does not directly target optimisation of product-to-user interaction performance, it improves query efficiency (R4 partly addressed by both replication strategies). Because of the low computation and message complexity of the approach as well as the incorporation of a replacement policy, the requirements R7 and R8 can be assumed to be supported by both replication strategies. However, neither of the two replication strategies addresses the requirements R2, R3, R5, and R6.

[26] Note that the question of whether any of the proposed strategies is able to cope with the scalability challenges of smart product systems is hardly answerable. For example, the scalability of a strategy is typically strongly affected by the underlying network architecture. Hence, this analysis can only provide an estimation of the scalability of related work that is to be detailed in future work.

3.1.3.3 PAST

PAST is a P2P-based distributed storage system presented in [DR02, RD01b] that operates on the structured P2P overlay network Pastry [RD01a] and aims at providing high levels of content availability, query efficiency, and scalability.

Past Replication. PAST consists of a replication strategy that improves content availability and persistence. When storing a content object in the network, the initiator defines the replication factor $k \leq \frac{l}{2} + 1$ with l denoting the maximum number of nodes in the leaf set as defined by Pastry. This way, the primary provider tries to creates $k-1$ replicas on the most suitable nodes of its leaf set. In order to balance available storage capacity among the nodes in the leaf set, PAST applies *replica diversion*. If any node i of the $k-1$ candidates mentioned above (i.e., *primary replica stores*) is not able to create a local replica given a pre-defined metric, it selects an alternative node j of its leaf set that is not one of the remaining $k-2$ candidates and that possesses the maximum available storage capacity. Then, i asks j to create a local replica and maintains a pointer to j (i.e., *diverted replica store*). To account for potential leave and fail events of i, another node h with the $i+1$ closest identifier also maintains a pointer to j. With regard to its impact on query efficiency, replica diversion is not applied recursively and the overall storage operation fails if any of the potential diverted replica stores declines to create a local replica.

PAST Caching. PAST provides a cache-based replication strategy to optimise query efficiency. PAST creates local replicas on intermediates that route content as part of storage and retrieval operations and possess enough available storage capacity. These *cached replicas* have lower persistence priority than replicas created by the strategy PAST Replication and can be removed at any time. For example, cached replicas are evicted if additional storage capacity is required to serve storage requests of content or replicas created by the strategy PAST Replication. The selection of cached replicas to be removed is managed by the GreedyDual-Size replacement strategy, which was originally designed for Web proxy caching [CI97].

Overall, PAST aims at optimising all content properties defined in Section 2.1.3 by two complementary replication strategies that are both claimed to achieve high levels of scalability (R8). The strategy PAST Replication enables configuration of the number of replicas per content (R5). However, it does not address any of the other requirements. For example, storage operations are aborted if the defined number of replicas cannot be placed. Even though this clearly improves content availability and persistence, it may not be applicable in resource-constrained smart product systems. In contrast, the cache-based replication strategy meets R7 by the applied replacement (i.e., removal) strategy and utilises temporal locality properties of access patterns. Moreover, although this approach does not optimise content access in the course product-to-user interaction directly, it aims at improving query efficiency (R4 partly addressed). All other requirements are out of scope of PAST's cache-based replication strategy.

3.1.3.4 Dynamo

Amazon's distributed key / value store Dynamo is a distributed database system that sacrifices consistency to cope with the enormous availability, scalability, and reliability requirements of Amazon's services. Dynamo is based on the structure P2P overlay network Chord that is extended by the concept of virtual nodes to account for node heterogeneity with regard to storage capacity [DHJ+07]. Similar to OceanStore, Dynamo applies a replication strategy in order to enhance content availability and persistence. When a primary provider initially stores a content object, it distributes $k-1$ replicas to nodes maintained in a content-specific preference list. Since replicas should be stored on $k-1$ distinct *physical* nodes (remember that Dynamo operates on Chord with virtual nodes), the preference list contains more than the $k-1$ successors of the primary provider in the hash space and skips positions in the latter. Moreover, the parameter k is not defined by the initiator but per Dynamo instance, which results in a uniform replication degree.

With regard to the requirements, Dynamo addresses the heterogeneity and scalability challenges of smart product systems (R7, R8), and enables minimal configuration per instance (R5 partly met). However, no other requirements are captured by Dynamo's replication strategy.

3.1.3.5 Replication Strategies proposed by [On04]

[On04] proposes a set of ranking-based dynamic replica placement heuristics for fully distributed, unstructured P2P systems that target fulfilment of requester-assigned availability requirements.

Local. The strategy *Local* lets requesters create local replicas after receiving requested content objects. However, as opposed to caching or similar threshold-based concepts, Local only creates a new replica in case the supplied availability is below the requester-specified target availability.[27] If requesters do not possess enough available storage capacity, LRU or LFU is used as replacement strategy.

Highly Up First. The replication strategy *Highly Up First* extends Local. Instead of having requesters creating local replicas, the latter rank themselves as well as their neighbours according to their uptime and place replicas on the highest-ranked node. This is based on the assumption that replicas placed on nodes that are longer connected to the network can potentially be reached by more requesters.

Highly Available First. The strategy *Highly Available First* is similar to Highly Up First. However, instead of ranking nodes according to their uptime, this strategy compares the supplied availability of nodes and places a replica on the highest-ranked node.

[27] A definition of *demand availability* and *supply availability* is presented in [On04].

Combined. Finally, the strategy *Combined* integrates the strategies Highly Up First and Highly Available First. Thus, Combine calculates the average values for both properties uptime and availability and places a replica on a node for which both properties exceed the calculated average. If multiple nodes feature uptime and availability above the calculated average, then Combine selects the "best" node using uptime as first and availability as second parameter.

The three replication strategies Highly Up First, Highly Available First, and Combined all required nodes to exchange knowledge required for making replication decisions. Yet, this additional effort is compensated by an increase of content availability compared to the strategy Local. While all strategies result in similar replication and replacement costs, an evaluation study revealed that the strategy Combined yields the highest level of content availability.

With regard to the investigated requirements, all four strategies address R1 and are assumed to approach R7 by means of the applied replacement strategy. Moreover, because of the low expected effort for exchanging node status information, requirements R8 can be assumed to be supported. Moreover, although knowledge of access patterns with geographic locality properties is not utilised, the three cooperative strategies Highly Up First, Highly Available First, and Combined tend to improve content availability for nodes belonging to a certain region by placing replicas on the region's most suitable node. Also, requester-side replication always implies enhanced query efficiency in case of access patterns with temporal and / or geographic locality properties. Yet, none of the requirements R3, R5, and R6 is approached.

3.1.3.6 Replication Strategies proposed by [MN05]

[MN05] proposes three mechanisms for estimating availability of individual nodes as well as groups of nodes with configurable look-ahead periods. These mechanisms are employed by three replication strategies that use availability prediction to optimise replica placement in terms of content availability.

MAS-x. *MAS-x* places replicas on k out of the first x of the primary provider's successors with the highest predicted availability.

sticky-rep. *Sticky-rep* places replicas on the first k of the primary provider's successors. However, as opposed to most structured P2P systems that maintain the invariant of storing content on nodes with identifiers being close to the content's identifier, sticky-rep does not perform any replacement in case of node churn affecting a provider's successor lists.

sticky-ap. *Sticky-ap* combines sticky-rep and MAS-x by placing replicas on k out of the first x of the primary provider's successors with the highest predicted availability without performing any re-organisation that might be caused by system dynamics.

Based on a simulation study, [MN05] demonstrates that replica placement on nodes with high expected availability leads to enhanced content availability as well as reduced re-organisation overhead. Consequently, sticky-ap achieves the highest level of content availability. However, in particular in resource-constrained systems, this approach may foster overloading of highly available nodes.

None of the requirements R1, R2, R4, and R6 are addressed by the proposed strategies. With regard to R3, the strategies MAS-x and sticky-ap could be attributed partial realisation. Even though they do not utilise any type of problem-solving knowledge, they apply an active concept that estimates future system state. Also, all three strategies provide some minimal configuration by means of the replication factor k (R5). Resource limitation and heterogeneity of smart products can be assumed to be addressed based on the reasonable supposition of a relation between node availability and node's resources (R7). Finally, despite the computation complexity of the availability predictors, the reduced re-organisation overhead positively affects overall system scalability (R8).

3.1.3.7 Replication Strategies proposed by [KRT06, KRT07]

Top-K. *Top-K* is an approach for enhancing content availability in structured P2P communities [KRT06]. Top-K considers the limited storage capacity of nodes by integrating LRU as replacement policy. If a node i receives a request from a requester r for a content object c that i is not a provider of even though it is (surrogate) root node, i leverages the overlay structure to determine the top $K-1$ candidates that might store a replica of the requested content c (i.e., the $K-1$ nodes whose node identifiers are closest to the requested content identifier). Node i queries all determined candidates for c. If any of these nodes stores the requested object c, i retrieves the latter, creates a local replica, and sends the result back to the original requester r. Otherwise, i retrieves c from outside the community, stores a local replica and sends the result to r. This leads to replication degrees following content popularity and enhances content availability within the P2P community.

Top-K Most Frequently Requested. The replacement policy LRU tends to store unpopular content on nodes even though this storage capacity could be better used by content with higher request rates. [KRT06, KRT07] approach this issue by the replication and replacement policy MFR. MFR lets each node maintain the access rate for all content objects it has been requested for within a configurable period of time and stores content with the highest ratios between access rate and object size, storing as many content as possible. This policy is combined with Top-K, resulting in the strategy *Top-K MFR*. Using this strategy, a requester r *sequentially* queries the top K candidates and stops as soon as it receives the requested content c. In case a candidate i is not a provider but should store c according to MFR, it retrieves the latter, creates a local replica, a sends the results to r. This yields a high level of availability as well as a nearly optimal number and placement of replicas by fostering higher replication degrees for popular content.

Both approaches utilise temporal locality properties of access patterns (R1) and provide means for configuration in terms of the parameter K (R5). Also, even though both strategies might lead to high replication degrees, they make use of replacement (i.e., removal) strategies to account for storage limitations of nodes (R7). Because of the purely local concept and the avoidance of maintenance overhead, it can be assumed that both approaches are highly scalable (R8). However, the strategies only address content availability and provide no means for enhancing query efficiency. They are purely reactive, do not make use of geographic locality properties, and have no possibility for applications to affect number and placement of replicas. Consequently, requirements R2 to R4 as well as R6 are not addressed.

3.1.3.8 Replication Strategies proposed by [She10]

The approach Integrated file Replication and consistency Maintenance (IRM) proposed by [She10] is a combined replication strategy for structured P2P overlay networks that takes into consideration consistency maintenance needs. IRM relies on *query initiating rates*, i.e., the number of queries per content object sent by a requester during a certain period, and *query passing rates*, i.e., the number of queries per content object received and passed on by an intermediate during a certain period, which are autonomously measures and analyses by each node. IRM consists of two replication strategies that make use of the afore-summarised parameters.

IRM Query Initiating Rates. If a requester r receives a requested content c and its query initiating rate for c exceeds a pre-defined threshold, then a local replica is created by the strategy *IRM Query Initiating Rates*.

IRM Query Passing Rates. The cooperative replication strategy *IRM Query Passing Rates* works as follows. If an intermediate i receives a request for content c for which (i) its query passing rate exceeds another pre-defined threshold and (ii) it possesses enough storage capacity for maintaining a local replica, then i extends the request with a replication request. If the actual provider receiving the request tends to become overloaded, it checks the request for additional replication requests and pushes replicas to the corresponding nodes. If there are no replication requests, then the provider pushes a replica to the node from which it received the most requests for c.

Moreover, in order to address the trade-off between query efficiency and consistency described in Section 2.1.1, IRM lets each node periodically re-assess its query initiating and passing rates, and remove replicas if any of the rates are below their thresholds. This way, IRM achieves high replica utilisation and optimises number and placement of replicas. This leads to a high level of query efficiency, fosters load distribution, and reduces the required consistency maintenance overhead.

With regard the to defined requirements, IRM relies on temporal locality properties of access patterns (R1) and enables configuration of the two threshold parameters (R5).

By means of periodic re-assessment and invalidation of replicas (removal policy), as well as its low complexity, IRM can moreover be assumed to address requirements R7 and R8. Requirement R4 can be rated as partly addressed. Even though IRM does not optimise content access related to product-to-user interaction, it aims at improving query efficiency. However, none of the requirements R2, R3, and R6 is supported.

3.1.3.9 Replication Strategies proposed by [LYC07]

[LYC07] proposes three erasure code replication strategies for improving content availability in unstructured, fully distributed P2P systems. The strategies vary in the amount of information used as well as their computation complexity. Moreover, all three strategies assume a static system (i.e., nodes and content to be stored per node are known) and do not consider diversity of communication delays or link bandwidth.

Random. The erasure code replication strategy *Random* consists of two phases. First, in the storage allocation phase, a provider determines the stretch factor based on the storage capacity available within its writable node set (i.e., nodes on which the provider can place erasure-coded blocks) as well as all content objects that are to be replicated within the latter (remember assumption of a static system). For this purpose, Random applies a locking-phase strategy to enable accurate, non-overlapping stretch factor estimation. Second, in the replica placement phase, for each content object to be stored, the provider places erasure-coded blocks on randomly selected nodes of its writable node set that possess enough available storage capacity. Hence, overall, the Random algorithm utilises the entire available storage capacity to place erasure-coded blocks. However, the random selection of nodes may lead to considerable variance of content availability subject to node availability.

Group Partition. The replication strategy *Group Partition* extends Random and approaches the above-stated variance by having nodes exchange availability information and by an adapted placement of erasure-coded block. Hence, instead of randomly selecting nodes out of the provider's writable node set, the latter is divided into k groups consisting of nodes with similar availability (k represents the number of erasure-coded blocks). Thereafter, the provider randomly selects one node from each group to perform block placement. This way, the difference in content availability is clearly reduced.

Highest Available First. The erasure code replication strategy *Highest Available First* aims at fulfilling a target level of content availability. The strategy consists of four phases. First, in the initialisation phase, a provider determines the stretch factor and calculates the average availability of nodes belonging to its writable node set as well as the average number of erasure-coded blocks per content in order to specify the target availability level. In the second phase, the nodes of the provider's writable node set are sorted in descending order of their availability. The content to be stored is split into b blocks (without erasure coding, i.e., the stretch factor is assigned a value of 1). These blocks

are virtually assigned to the b highest available nodes of the writable node set and the resulting content availability is calculated. Thereafter, in phase three, the calculated content availability is compared with the target availability and the erasure code redundancy is increased incrementally until the target availability is met. Finally, the erasure-coded blocks are created and placed on the nodes determined afore (phase 4). This way, the content-specific erasure code redundancy is proportional to the availability of the target nodes. Especially if the overall shared storage capacity is very limited and if there are only few nodes with high availability, the Highest Available First strategy outperforms both Random and Group Partition in terms of content availability.

Interestingly, although the proposed strategies generate high levels of content availability and persistence, they support none of the defined requirements but storage capacity (R7) and scalability (R8). Storage capacity is implicitly captured in the replica placement phase of the proposed algorithms, while their scalability can be assumed because of the low amount of exchanged information.

3.1.3.10 Summary

The classification of the analysed P2P replication strategies based on the scheme described in Section 3.1.2 is presented in Table 3.2. The strategies are discussed again in Section 3.2, which – for each strategy – summarises the fulfilment of the requirements presented in Section 2.4 in order to identify and assess the work closest to the content placement strategies proposed in this thesis.

3.1.4 Content Distribution Networks

This section summarises replication strategies used in CDNs. A survey of CDN replication strategies is presented in [KKM02].

3.1.4.1 Mobile Dynamic Content Distribution Network

The Mobile Dynamic Content Distribution Network (MDCDN) proposed by [AMAM04] solves the dynamic content placement problem for mobile CDNs. In order to minimise total communication costs (i.e., replication creation costs, replica maintenance costs, request forwarding costs), MDCDN nodes monitor and forecast varying user demand to determine whether to create new replicas or remove existing ones. For this purpose, MDCDN nodes apply the statistical demand forecasting method *double exponential smoothing* as part of the two active replication strategies *MDCDN Direct Demand* and *MDCDN Indirect Demand*. This prediction enables nodes in MDCDN to dynamically replicate content that features high probability of being requested in upcoming periods, thereby

Replication Strategy	Replica Granularity	Replication Degree	Replication Schedule	Replication Initiation	Replica Placement	Sphere of Knowledge	Activity History
OS Deep Archival Storage [KWZ+00]	Block-Level	Entity-Specific	On-Demand	Provider	Metric	Local	Reactive
OS Put [KWZ+00]	Partial	Uniform	One-Time	Provider	Metric	Local	Reactive
OS Introspective Replica Mgmt [KWZ+00]	Partial	Entity-Specific	Periodic	Provider	Provider-Side	Cooperative	Active
OS Caching [KWZ+00]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Reactive
OS TTL-based Replication [Gee02]	Partial	Entity-Specific	On-Demand	Intermediate	Metric	Local	Reactive
Freenet Insert [CSWH01, ATS04]	Partial	Entity-Specific	One-Time	Initiator	Path Initiator-Provider	Local	Reactive
Freenet Request [CSWH01, ATS04]	Partial	Entity-Specific	On-Demand	Requester	Path Provider-Requester	Local	Reactive
PAST Replication [DR02, RD01b]	Partial	Entity-Specific	One-Time	Initiator	Provider-Side	Local	Reactive
PAST Caching [DR02, RD01b]	Partial	Entity-Specific	On-Demand	Initiator, Requester	Path Initiator-Provider, Path Provider-Requester	Local	Reactive
DRP Local [On04]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Reactive
DRP Highly Up First [On04]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side, Metric	Cooperative	Reactive
DRP Highly Available First [On04]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side, Metric	Cooperative	Reactive
DRP Combined [On04]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side, Metric	Cooperative	Reactive
MAS-x [MN05]	Partial	Uniform	One-Time	Provider	Provider-Side, Metric	Cooperative	Active
sticky-rep [MN05]	Partial	Uniform	One-Time	Provider	Provider-Side	Local	Reactive
sticky-ap [MN05]	Partial	Uniform	One-Time	Provider	Provider-Side, Metric	Cooperative	Active
Top-K [KRT06]	Partial	Entity-Specific	On-Demand	Intermediate	Provider-Side	Local	Reactive
Top-K Most Freq. Requested [KRT06, KRT07]	Partial	Entity-Specific	On-Demand	Intermediate	Provider-Side	Local	Reactive
ECR Random [LYC07]	Block-Level	Uniform	One-Time	Provider	Provider-Side	Local	Reactive
ECR Group Partition [LYC07]	Block-Level	Uniform	One-Time	Provider	Provider-Side, Metric	Cooperative	Reactive
ECR Highest Available First [LYC07]	Block-Level	Entity-Specific	One-Time	Provider	Provider-Side, Metric	Cooperative	Reactive
Dynamo [DHJ+07]	Partial	Uniform	One-Time	Provider	Provider-Side	Local	Reactive
IRM Query Initiating Rate [She10]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Reactive
IRM Query Passing Rate [She10]	Partial	Entity-Specific	On-Demand	Provider	Path Provider-Requester	Cooperative	Reactive

Table 3.2: Classification of P2P Replication Strategies

reducing latency of future requests (*direct demand*). In addition to replicating content on the requester-side for addressing direct demand, MDCDN pushes replicas to nodes that cause high predicated *indirect request rates* in order to avoid overloading. A detailed description of the algorithms used by MDCDN is presented as part of the evaluation study in Section 5.3.6.1.

Both replication strategies apply a prediction method that incorporates temporal locality properties of access patterns (R1 partly addressed). In contrast, access patterns with geographic locality properties, are not taken into account (R2). Also, even though MDCDN does not use problem-solving knowledge of smart products, the active approach partly meets requirement R3. Content pre-replication according to estimated direct demand can be assessed as an indirect addressing of requirement R4. As presented in Section 5.3.6.1, MDCDN provide various means for configuration (R5). While the recursive calculation of MDCDN's prediction method lead to high computation complexity, the purely local nature of the strategy does not require any information exchange (R8 partly addressed). Finally, although MDCDN does not take into consideration resource limitations of nodes, the authors reasonably state that this could be integrated easily (R7 partly addressed). Means for enabling applications to affect number and placement of replicas are not captured (R6). Overall, MDCDN appears to be applicable to the domain of smart products. Also, if the computation complexity represented an issues, one could think of an operating model that lets MDCDN runs on dedicated nodes, only.

3.1.4.2 Replication Strategies proposed by [On04]

In addition to the dynamic replication strategies for P2P systems presented in Section 3.1.3.5, [On04] proposes as set of ranking-based heuristics for the static replica placement problem in CDNs (i.e., system state and access patterns are assumed to be known). These heuristics apply full replication and aim at determining number and placement of replicas to achieve a defined target level of content availability.

Highly Available First. The strategy *Highly Available First* distributes a pre-defined number of k replicas on the nodes with the highest availability. Given the assumption of a known system state, node availability can be determined easily. Compared to a random placement, this yields increased average and minimum content availability but slightly reduces query efficiency, which [On04] measures in terms of the number of hops a request passes until it gets served.

Transit Nodes. Based on the assumption that the higher the number of communication links of a node the more nodes it can reach with low latency, the strategy *Transit Node* places a number of k replicas on nodes with most connections. While Transit Node fails to reach the level of content availability achieved by Highly Available First, it improves the latter in terms of query efficiency.

COM. The strategy *COM* combines Highly Available First and Transit Nodes. It builds an ordered candidate set of nodes with availability values and number of communication links above average and selects the k highest ranked nodes with node availability being used as primary criterion. Due to the focus on node availability, COM only yields slight improvement over Highly Available First in terms of content availability and query efficiency.

Admission Control. While the above-described strategies try to achieve a given target level of content availability, the strategy *Admission Control* aims at guaranteeing request-assigned availability needs as well as at improving query efficiency. The strategy applies admission control to determine whether to maintain or revoke replicas. Moreover, admission control is used to assign requests to providers in a way that requester-assigned availability requirements are fulfilled without overloading providers taking into account bandwidth and storage capacity constraints. Admission Control is made up of three procedures.

First, the procedure *highest available path* is used to determine a replica placement plus an assignment of requests to providers that meets content availability requirements of all requests. For this purpose, given that failure probabilities are known, the procedure identifies the path with the highest availability (i.e., minimum failure probability) given a configurable maximum number of overlay hops for each request. Thereafter, the procedure assesses whether the highest available paths fulfil the target availability of the requests. For each path that does not meet this requirement, a replica is placed on a

node close to the provider that possesses enough resources and is able to guarantee the defined availability objective.

Second, after a replica placement with content availability guarantees is identified, the procedure *delete and merge* is used to reduce the overall number of replicas. For each provider, the procedure checks whether the assigned requests could be served by another provider without violating content availability. If so, the replica is deleted from the provider.

Finally, the procedure *move and update* aims at enhancing query efficiency. For each provider, the procedure examines whether any of the provider's neighbours could enhance query efficiency while preserving content availability of all assigned requests. If so, the replica is moved from the provider to the identified neighbour. Overall, this results in an optimised number and placement of replicas that guarantees content availability of all requests.

The proposed strategies – in particular because of their assumption of system state and access patterns to be known – address only few of the defined requirements. All strategies take into consideration node storage capacity (R7). Also, Transit Node, COM, and Admission Control tend to improve content availability and query efficiency in case of access patterns with geographic locality properties (R2 partly addressed). However, even though global knowledge could be predicted periodically based on collaborative observations, centralised approaches do in general not cope with the challenges of smart products.

3.1.4.3 Replication Strategy proposed by [HA04]

[HA04] proposes a replication strategy that aims at determining minimum number and placement of replicas for achieving target access delay guarantees. As opposed to the strategies presented in [On04], this work defines the term *guarantee* in a "loose sense, to mean trying to achieve latency bounds with a 'high probability' "[HA04, p. 1]. The approach enables users to negotiate access latency bounds with CDN service providers by means of Quality of Service (QoS) contracts. *Latency* is defined as the time needed for serving requests within the CDN that cannot be served directly by the node receiving the request. *Latency bounds* are formulated as a step function subject to the content size. This enables content classification and eases replication decisions, which can be made per content class instead of on the level of individual content objects.

[HA04] formulates the replication problem as a *graph domination problem* and proposes an algorithm for periodically solving this problem in fully distributed systems. The algorithm assumes that each node n is aware of all nodes that can query n for content of a certain class within the class-specific latency bound (i.e., the *span* of node n including all requesters it can cover). Moreover, the algorithm supposes that each node n knows all nodes it can request content from without violating the class-specific bound. On this

basis, the strategy lets all nodes send their span information to all nodes covered by them. Thereafter, for each class, the nodes cooperatively determine the *dominating set*. In case a node cannot be covered by any other node, it needs to be part of the dominating set and announces its participation to all nodes it covers. Else, the node analyses the span information it received, determines the node with the highest span, and notifies this node to become member of the dominating set. A notified node enters the dominating set, announces its participation to all nodes it covers, and stops nomination of other nodes. For each content-class, the resulting content-class-specific dominating set contains the nodes on which to place replicas of content belonging to the class in order to achieve the class-related latency bounds.

Since the algorithm does not employ any means of voting or synchronisation, it is very effective for approaching the graph domination problem in a decentralised way. Moreover, due to the low amount of information consumed by the strategy, both requirements R7 and R8 can be assumed to be met. The negotiation of target latency bounds provides applications with means for affecting number and placement of replicas (R6). Furthermore, although the approach does not aim at improving performance of product-to-user interaction, simulation results reveal that it is capable of achieving different pre-defined levels of query efficiency with high confidence (R4 partly addressed). Yet, the requirements R1, R2, R3, and R5 are not met by the proposed strategy. Also, the dynamics of smart product systems may pose the additional challenge of how to configure the period in which replica organisation is to be re-assessed in order to achieve given access latency bounds.

3.1.4.4 Replication Strategy proposed by [BB04]

[BB04] proposes the strategy Cost-Quality Optimised Replica Placement (CQORP) for adaptive CDNs, which targets adaptation and replication of content representations depending on device capabilities and bandwidth restrictions in ubiquitous computing environments. For this purpose, CQORP targets profit maximisation taking into consideration adaptation and transmission costs as well as the quality of a content representation from the perspective of the requester measured in terms of revenue. CQORP takes into account storage limitations and assumes knowledge about system load and state.

Given the above objective, [BB04] defines an algorithm for determining the adaptation path with the highest profit for each given request. For this purpose, the authors propose an *adaptation path graph* that is modelled as a weighted graph with the weighting representing adaptation operation costs. The most suitable path is determined by applying traditional shortest path search algorithms (shortest path yields maximum profit). Second, two ranking heuristics (plain and greedy) for the capacity-constrained replica placement problem are proposed. Both heuristics use the profit gained by a replica given a certain reference placement weighted by the size of the replica as benefit function.

Replication Strategy	Replica Granularity	Replication Degree	Replication Schedule	Replication Initiation	Replica Placement	Sphere of Knowledge	Activity History
MDCDN Direct Demand [AMAM04]	Partial	Entity-Specific	Periodic	Requester	Requester-Side	Local	Active
MDCDN Indirect Demand [AMAM04]	Partial	Entity-Specific	Periodic	Provider	Path Provider-Requester	Local	Active
SRP Highly Available First [On04]	Full	Uniform	One-Time	Central Entity	Metric	Local	Active
SRP Transit Node [On04]	Full	Uniform	One-Time	Central Entity	Metric	Local	Active
SRP COM [On04]	Full	Uniform	One-Time	Central Entity	Metric	Local	Active
SRP Admission Control [On04]	Full	Entity-Specific	One-Time	Central Entity	Metric	Local	Active
Latency Guranatees [HA04]	Partial	Entity-Specific	Periodic	All Nodes	Metric	Cooperative	Reative
CQORP [BB04]	Partial	Entity-Specific	One-Time	Central Entity	Metric	Local	Active

Table 3.3: Classification of CDN Replication Strategies

While the greedy ranking provides better results (the benefit of a replica depends on the placement of other replicas), it comes with the cost of higher computation complexity.

The strategy takes into consideration resource limitation (R7) and could be classified to partly meet R4 by providing nodes with content in its most suitable representation. The requirements R1, R2, R3, R5, R6, and R8 are not addressed by CQORP. In conclusion, even though the idea of combining content adaptation and content replication appears to be an important topic in the area of the ubiquitous computing, it does not approach the specific needs formulated in this thesis (see Sections 2.2 and 2.4). Most important, the proposed solution is designed to be executed by a central entity having global knowledge about system state and load. Even though the authors present some initial thoughts on how the strategy could be extended to be applicable in dynamic systems, centralised approaches do typically not achieve high levels of scalability.

3.1.4.5 Summary

The classification of the analysed CDN replication strategies based on the scheme described in Section 3.1.2 is presented in Table 3.3. However, only MDCDN and the latency guarantee approach presented by [HA04] appear to be applicable in the domain of smart products (see discussion of closest related work presented in Section 3.2).

3.1.5 Grid Computing

This section summarises replication strategies used in Grid computing environments. A survey of replication strategies in this domain is presented in [ASD12].

3.1.5.1 Replication Strategies proposed by [RF01]

Best Client Replication. [RF01] proposes the replication strategy *Best Client*, which targets optimisation of content locality in order to improve query efficiency and bandwidth consumption in the domain of high-performance data grids. Best Client lets each provider periodically analyse indirect request rates for all content stored on-board. If an indirect request rate exceeds a pre-defined threshold, then the provider places a replica on the node that requested the content most often. Moreover, the indirect request rate for the replicated content is cleared to enable unbiased assessment in upcoming periods taking into account potential dynamics of access patterns. The replication strategy is complemented by a replacement (i.e., removal) strategy that combines content popularity with the time for which content is stored on-board. Thus, if there are multiple candidates with the same degree of popularity, the one that is longest stored on-board is removed.

By placing replicas on the nodes with the highest observed requests in a given period, Best Client indeed assumes and utilises temporal locality properties of access patterns (R1). Also, the approach is very lightweight and can be assumed to achieve high levels of scalability (R8). The incorporation of a replacement strategy moreover addresses the storage limitation captured by requirement R7. Finally, requester-side replica placement positively affects query efficiency (partial realisation of R4). All other requirements are not met by the proposed strategy Best Client.

Cascading Replication. In addition to Best Client, [RF01] proposes the replication strategy *Cascading*, which is tailored towards the hierarchical structure of multi-tier data grids. Cascading assumes all content to be initially stored at the root node of the hierarchy. Similar to Best Client, the root node uses a threshold-based concept to replicate popular content. Yet, in contrast to Best Client, a replica is not placed on the leaf node (i.e., the requester) but on the child of the root node that belongs to the path to the client with the highest request rate. This concept is applied by all providers (initially the root node is the only provider), which may result in a popular content being eventually replicated to leaf nodes.

The main advantage of this approach is the incorporation of geographic locality properties in addition to temporal locality properties (R1 and R2). Moreover, Cascading employs the replacement policy used by Best Client to account for resource limitations of nodes (R7) and can be assumed to be highly scalable (R8). Also, similar to Best Client, query efficiency is improved in the long term (R4 partly addressed). [RF01] moreover proposes a combination of Cascading with requester-side caching to further improve query efficiency. However, none of the requirements R3, R5, R6 is addressed and the general applicability of the strategy is reduced by the reliance on hierarchical structures.

Fast Spread. [RF01] defines the replication strategy *Fast Spread*. Fast Spread replicates content along the path from provider to requester and employs the replacement

policy used by Best Client and Cascading. Hence, Fast Spread addresses the same requirements as Cascading but leads to a strong increase in the overall number of replicas. Also, the concept could be applied to non-hierarchic structures. This would lead to an approach similar to the request-based replication strategy of the P2P storage system Freenet (see Section 3.1.3.2).

3.1.5.2 Replication Strategy proposed by [CDL+07]

In order to improve workflow execution performance in Grid computing environments, [CDL+07] proposes a decoupling of the control flow and the data flow of workflows as well as an asynchronous pre-replication of content required during workflow execution. The authors assume non-branched worflows (i.e., neither OR nor XOR transitions) as well as a workflow management system being aware of activity-related content needs (e.g., by means of annotations). The proposed *workflow-based replication strategy* utilises knowledge of upcoming workflow-related content demand explicitly provided by the workflow management system. This leads to a reduced need for on-demand content retrieval as well as an enhancement of workflow execution performance. Moreover, based on an experimental setup, [CDL+07] demonstrates that the value of asynchronous data staging increases with the amount of workflow-related content needs.

While the proposed strategy does not utilise temporal or geographic locality properties of access patterns in makes use of and fully relies on knowledge about upcoming workflow-related demand (R3). Also, the applied pre-replication significantly improves query efficiency and – consequently – workflow execution performance. However, since the workflows in focus do typically not include product-to-user interaction, requirement R4 is only partly addressed. Moreover, the strategy appears to be highly scalable because of its purely local concept (R8). Requirements R5, R6, and R7 are not addressed.

3.1.5.3 Replication Strategy proposed by [YWD10]

Best Reply proposed by [YWD10] is a Nash-equilibrium-based replication strategy for data grids and is integrated with a decentralised task scheduling algorithm.[28] The strategy aims at improving task execution performance by optimising number and placement of replicas. It is operated by all nodes and maintains request rates retrieved within a specific period of time.[29] The replication procedure is triggered in case the request rate of a content not stored on-board exceeds a pre-defined threshold within a period. If so, each node for which the condition applies individually compares the benefit of the current set of content stored on-board with the set after creating a local replica of the content

[28] This term *task* used in this work corresponds to the term *workflow* described in Section 2.1.5 (note that a workflow could also consist of a single activity, only).

[29] Note that the system model used by Best Reply assumes requests to be broadcasted to all nodes.

under consideration. Also, potential removal of content stored on-board is taken into account by means of an LRU replacement strategy. The actual benefit calculation applies game theory concepts as a function of access latency between requester and (potential) provider. The authors prove that the Best Reply algorithm leads to the Nash equilibrium for the system as a whole.

The threshold-based replication strategy assumes access patterns with temporal locality properties (R1) and targets improvement of query efficiency (R4 partly addressed). It takes into account resource limitation by an LRU-based replacement strategy and can be assumed to achieve high levels of scalability given its decentralised and local concept (R7, R8). However, even though the strategy is built upon tasks, it does not utilise task-related content needs to actively enhance replica organisation (R3). Also, none of the requirements R2, R5, and R6 is addressed.

3.1.5.4 Replication Strategy proposed by [KIS11]

PHFS proposed by [KIS11] is an active replication strategy for multi-tier data grids that aims at enhancing query efficiency. PHFS extends Fast Spread presented in Section 3.1.5.1 with a pre-replication strategy that makes use of data mining techniques to predict future demand. Based on the design of multi-tier grids (the higher a node is positioned in the hierarchy, the more resources it possesses), PHFS applies *hierarchic replication*. This means, the number of replicas decreases with the depth of the path from root to client in order to achieve optimal resource utilisation as well as to enhance content locality.

PHFS consists of two stages. In stage 1, the prediction is performed centrally by the root node when it receives the first request for a certain content object. The root node inquires access information from all other nodes and applies data mining techniques to infer knowledge such as content clusters or sequential access patterns. The inferred relationship between the requested content and other objects is weighted. Objects with a relationship exceeding a pre-defined threshold are organised in a so-called Predictive Working Set (PWS), in which they are sorted in descending order of their quantified relationship. The resulting PWS is assigned an initial priority by the root node and is attached to the requested content. Hence, content objects being organised in a PWS share spatial locality and it is assumed that each PWS captures a certain degree of geographic and temporal locality.

In stage 2, for each PWS, the replication configuration is adapted by all nodes in a distributed way based on local knowledge. This adaptation consists of three modules. First, the *PWS recording module* enriches each PWS with local knowledge. Second, the *priority change module* makes use of the added knowledge to periodically adapt the priority of each PWS subject to request rates expected in the upcoming period. Third, the *replication configuration change module* makes use of the updated PWS priorities to periodically

Replication Strategy	Replica Granularity	Replication Degree	Replication Schedule	Replication Initiation	Replica Placement	Sphere of Knowledge	Activity History
Best Client Replication [RF01]	Partial	Entity-Specific	Periodic	Provider	Requester-Side	Local	Reactive
Cascading Replication [RF01]	Partial	Entity-Specific	Periodic	Provider	Metric	Local	Reactive
Fast Spread [RF01]	Partial	Entity-Specific	Periodic	Provider	Path Provider-Requester	Local	Reactive
Workflow-based Replication [CDL+07]	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Active
Best Reply [YWD10]	Partial	Entity-Specific	Periodic	All Nodes	Metric	Local	Reactive
PHFS [KIS11]	Partial	Entity-Specific	Periodic	All Nodes	Path Provider-Requester	Cooperative	Active

Table 3.4: Classification of Grid Computing Replication Strategies

adapt number and placement of replicas given pre-defined upper and lower thresholds. If a PWS priority falls below the lower threshold, the least used PWS-related content objects are flagged as replacement candidates. Similarly, if a PWS priority exceeds the upper threshold, then the content objects of the PWS with the highest requests rates are replicated and placed on lower-layer nodes. If required, replacement candidates or least recently accessed content objects of the PWS with the lowest priority are replaced to free the necessary storage capacity. Moreover, if content objects of a certain PWS have higher access rates than objects of the same PWS in lower layers, then these objects are exchanged in order to place the most popular content as close to the requesters as possible.

This leads to increased locality of content and enhanced query efficiency (R4 partly addressed). With regard to the remaining requirements, PHFS addresses R1 and R2 as well as R7 by means of an integrated replacement policy. Also, requirement R3 can be classified as partly met because of the active replication approach. The application of data mining algorithms as well as the amount of information exchanged between nodes may reduce the overall scalability of the strategy, in particular given the resource limitation of smart products (R8 party addressed). Requirements R5 and R6 are not approached. However, most important, PHFS is tailored towards hierarchic system structures and requires access patterns with some degree of spatial locality in order to achieve its full potential. This clearly reduces its general applicability in the domain of smart products.

3.1.5.5 Summary

The classification of the analysed replication strategies in the domain of Grid computing based on the scheme described in Section 3.1.2 is presented in Table 3.4. Note that Cascading replication (see Section 3.1.5.1) and PHFS (see Section 3.1.5.4) do not appear to be generically applicable in the domain of smart products, because of their reliance on hierarchic system structures. The strategies are discussed again in Section 3.2, in order to identify and assess the work closest to the content placement strategies proposed in this thesis.

3.1.6 Client / Server Storage Systems

3.1.6.1 Replication Strategy proposed by [BWYH06]

[BWYH06] proposes an architecture for enhancing mobile information access in occasionally connected computing environments. The proposed *worklet-based replication strategy* makes use of workflows, user profiles, as well as environmental information in order to predict and proactively collect and aggregate content that is most likely required by mobile devices in the upcoming period. To enable modelling of content needs, the strategy utilises so-called "worklets", which represent self-contained workflows combined with execution rules. Due to the limited processing capability of mobile devices, the actual prediction and pre-replication is processed by a central data staging server. By "charging" mobile devices in periods of strong connection with content that is likely needed in the nearby future, the proposed strategy aims at enhancing query efficiency and content availability.

The proposed replication strategy utilises "worklets" for actively optimising number and placement of replicas and improving both availability and query efficiency of information accessed by the user.[30] Hence, requirements R3 and R4 are clearly met. Also, the approach takes into account resource limitations (R7). However, none of the requirements R1, R2, R5, and R6 is addressed and the centralised architecture cannot be assumed to cope with the scalability challenges in the domain of smart products (R8).

3.1.6.2 Replication Strategy proposed by [DBEN07]

[DBEN07] proposes an erasure-code replication strategy that aims at smoothing the rate at which new erasure-coded blocks are to be created, i.e., the repair rate, while ensuring high levels of content availability and persistence given node churn. The proposed active strategy applies erasure-code replication, which is combined with a periodic estimation of node failure behaviour to anticipate the number of repair blocks to be created. The estimation is realised based on a mathematical model that captures node behaviour as a *Continuous-Time Semi-Markov* chain and the overall system as a network of two $G/G/\infty$ queues.

The approach is based on the assumption that (i) the number of available erasure-coded blocks is known and (ii) erasure-coded blocks can be stored on distinct nodes, i.e., node availability implies block availability. On this basis, the predictor estimates the number of nodes that permanently leave the network in the upcoming period in order

[30] In the mobile scenarios presented in the following sections, content being pre-replicated is expected to be accessed by the user. Hence, optimisation of query efficiency directly affects product-to-user interaction as stated in requirement R4.

to infer the number of erasure-coded blocks to be created. Moreover, to account for system dynamics, the inter-estimation time depends on the number of disconnections instead of using a fixed period length. This means, the estimation procedure is executed after a certain number of permanent disconnections are observed. Finally, the proposed active strategy is complemented by a reactive strategy that is automatically applied in case the number of available erasure-coded block meets a configurable lower threshold. This *hybrid strategy* may result in peeks in system load, which are stated by the authors as indispensable for ensuring availability and persistence.

The authors assume the existence of an entity that is able to monitor block availability and make replication decisions without tailoring the proposed strategy to any specific system architecture.[31] However, in either case, the scalability of the strategy can be assumed to be limited. On the one hand, centralised structures typically face scalability issues. On the other hand, if the strategy was realised in a distributed way, its scalability would likely be affected by the high amount of information to be exchanged in order to obtain an overall knowledge about the number of available erasure-coded blocks. Hence, requirement R8 is denoted as partly addressed. Also, R3 and R5 are partly met because of the active component of the replication strategy and its configuration options, respectively. All other requirements are not supported.

3.1.6.3 Replication Strategy proposed by [SGAM11]

[SGAM11] proposes a mechanism to enhance user-perceived efficiency of mobile access to business data maintained in enterprise information systems. The mechanism applies pre-replication by combining *sequence prediction* with a *response piggybacking* approach. Triggered by a request from a mobile device, the Sequence Prediction Algorithm (SPA) FxL presented in Section 3.1.7.3 is applied at the level of a proxy server interconnecting mobile devices and enterprise information systems in order to estimate future demand (in the given setup: the next demand / request). The predicted content is piggybacked with the actual response and pre-replicated to the device. In order to reduce pre-replication wastage, content piggybacking is only applied if (i) the demand prediction achieves a pre-defined confidence level and (ii) the predicted content is uncritical (e.g., in terms of consistency). Moreover, to continuously enhance prediction accuracy, the SPA is informed about the actual utilisation of pre-replicated content.

Evaluation results reveal that the proposed strategy has a positive effect if prediction accuracy exceeds 15%, which can be assumed for FxL. The actual performance gain depends on network latency (higher latency increases performance gain) and network bandwidth (higher bandwidth increases performance gain).

[31] Note that because of this assumption, the strategy is presented in the section on client / server storage systems.

The proposed replication strategy does not explicitly utilise temporal and geographic locality of access patterns. Instead, it applies an SPA to predict future demand (R3 partly addressed). Moreover, requirements R4 and R5 are met by the piggybacking approach and the configuration options of FxL, respectively. Resource limitation is taken into account by limiting pre-replication to a single object (R7). However, neither of the requirements R6 and R8 (because of the centralised architecture) are approached.

3.1.6.4 Replication Strategy proposed by [JIR+09]

[JIR+09] proposes a *combined replication and replacement strategy* the targets improvement of query efficiency of content associated with location-based services in mobile environments. The strategy supposes that requests follow a certain sequence and that successively accessed location-based services are geographically close. Based on this assumption, an *association rule mining* mechanism is applied on a central node to predict the next location-based service; the content related to this service is pre-replicated to the mobile device.

This pre-replication concept is complemented by an invalidation mechanism named *dual valid scopes*. Given the application domain, a valid scope is defined as a "geographic region in which a data item value is valid" [JIR+09, p. 2]. Valid scopes are modelled by means of Voronoi diagrams [OBS+08] in the form of exact polygon endpoint schemes and approximated circle schemes. Moreover, node storage is divided into three parts: the *primary part* to store explicitly requested content, the *secondary part* for pre-replicated content, and the *tertiary part* for "aged" pre-replicated content. Pre-replicated content is moved from secondary to primary storage after being requested as well as from secondary to tertiary storage if it has not been accessed within a configurable period of time. This partitioning enables differentiated replacement strategies that take into consideration content status (e.g., explicitly requested content is not replaced for pre-replicated content).

[JIR+09] propose a tripartite replacement (i.e., removal) strategy that takes into account content distance (i.e., "the distance between the current location of a mobile client and the reference point of a valid scope for a data value" [JIR+09, p. 4]), content age (i.e., time passed since content was last accessed), and access probability calculated by means of an exponential aging method. First, for the primary storage, the replacement strategy applies all of the above criteria plus the invalidation mechanism dual valid scopes modelled in the form of an exact polygon endpoint scheme. Second, for the secondary storage, the replacement strategy employs content age, only. Finally, the replacement strategy of the tertiary storage utilises content age, content distance, and dual valid scopes modelled in the form of a less accurate circle scheme.

The proposed strategy addresses requirements R4 and R7 by means of requester-side pre-replication and the integrated replacement strategy, respectively. Also, due to its ac-

Replication Strategy	Replica Granularity	Replication Degree	Replication Schedule	Replication Initiation	Replica Placement	Sphere of Knowledge	Activity History
Worklet-Based Replication [BWYH06]	Partial	Entity-Specific	On-Demand	Central Entity	Requester-Side	Local	Active
Hybrid Replication [DBEN07]	Block-Level	Uniform	Periodic	Central Entity	Metric	Cooperative	Active
Piggybacking in Mobile Scenarios [SGAM11]	Partial	Entity-Specific	On-Demand	Central Entity	Requester-Side	Local	Active
Cache Prefetch and Replacement [JIR+09]	Partial	Entity-Specific	On-Demand	Central Entity	Requester-Side	Local	Active

Table 3.5: Classification of Client / Server Replication Strategies

tive concept, the strategy partly meets requirement R3. Yet, none of the requirements R1 and R2 as well as R5, R6, and R8 (because of the centralised architecture) is addressed.

3.1.6.5 Summary

The classification of the analysed replication strategies in the domain of client / server storage systems based on the scheme described in Section 3.1.2 is presented in Table 3.5. Note that the strategy proposed by [JIR+09] does not appear to fit the domain of smart products, because of its centralised prediction concept. All other concepts might be applicable with some adaptation and are further discussed in Section 3.2.

3.1.7 Content Access Predictors

As stated in Section 3.1.2, active replication strategies employ different kinds of predictors to estimate upcoming demand and pre-replicate content, accordingly. While the strategies presented in the previous sections already cover several prediction mechanisms, this section focuses on SPAs. In general, SPAs aim at predicting the next symbol given an input sequence of already observed symbols of varying length. They typically result in a set of probabilities for each potential symbol to appear next. According to the classification presented in [PSMS10], this section presents an overview of static and dynamic SPAs. In addition, the two user action predictors FxL and AFxL used by the active replication strategy presented in Section 3.1.6.3) are described.

3.1.7.1 Static Content Access Predictors

Static content access predictors represent the most basic approach for estimating upcoming demand. Even though these predictors employ static heuristics only and do not adapt their strategy to changes in the access pattern, some of them are known to be very effective.

First Successor (FS) uses the first-observed successor as the general, constant prediction [AL01]. FS is extended by *First Stable Successor (FSS)*, which requires a successor to be

observed N times in a row, before it is used as constant prediction [SPAL04]. In contrast to the fixed prediction of FS and FSS, *Last Successor (LS)* maintains the last-observed successor and is – consequently – sensitive towards variation in content access [LD97]. *Stable Successor (SS)* proposed by [AL01] approaches this sensitivity. It extends LS by using the last successor as prediction if it was observed for at least N times consecutively. If there is no such content, then no prediction is made. This enables balancing of the trade-off between prediction rate and reliability.

SS is further extended by *Recent Popularity (RP)*, which predicts the content as successor that occurred in at least j out of the last k requests. If there are multiple candidates, RP builds upon temporal locality properties and selects the candidate that was accessed most recently. If no such content exists, then no prediction is made [ALPB02]. Finally, *Predecessor Position (PP)* and *Pre-Predecessor Position (PPP)* base their prediction on successors of sequences of two and three requests, respectively [WPA+03].

3.1.7.2 Dynamic Content Access Predictors

Dynamic content access predictors are able to adapt their prediction strategy to changes in the access pattern. In fact, the suitability of any predictor depends on the given access pattern and strategies combining multiple predictors most often outperform single predictors in systems with changing workload.

The *Composite Predictor* proposed by [WPA+03] employs the four static predictors SS, PP, PPP, and RP. Based on simulation studies with various access patterns, each predictor is assign a weighting factor denoting its accuracy and the strategy dynamically selects the predictor with the highest probability of being correct. *Multiple Expert* maintains each successor occurred in past periods as expert, which is weighted according to its accuracy over time. This way, the highest weighted expert (i.e., successor) is used as prediction [Bra04].

[RP05] proposes a *perceptron-based approach* that does not limit prediction to a single successor but predicts several upcoming requests at once. Finally, [PSMS10] proposed a prediction strategy that combines the three static predictors LS, SS, and RP, and utilises a *neural network* to determine the most suitable strategy. For maintaining a high level of prediction accuracy, the strategy makes no prediction if the probability of correctness provided by the neural network is below a pre-defined threshold for all static predictors.

3.1.7.3 User Action Predictors

[HS07] proposes the two SPAs *FxL* and *AFxL* that target estimation of upcoming user actions in order to facilitate proactive user interfaces. An adaptation towards prediction of future content access is presented in [SGAM11].

FxL and AFxL are based on the KO algorithm defined by [KOS04] and combine results from different order Markov models. The algorithms utilise n-gram tries of frequencies of varying input sub-sequences in order to infer scores for each symbol representing the probability of its occurrence. The actual score calculation combines frequencies with a weighting factor, which differs for the two proposed SPAs. First, FxL applies a weighting factor equal to the length of the matching suffix of the input sequence (longer suffix yields more reliability). Second, AFxL takes into account the prediction quality of the applied order models by means of the rate at which they provided accurate predictions.

Both SPAs support configuration of the maximum sequence length, the minimum confidence level, and the prediction reach [SGAM11]. Evaluation results reveal that the simple strategy FxL outperforms AFxL in terms of accuracy as well as the size of the data structure.

3.2 Discussion

This section summarises the extent to which the requirements defined in Section 2.4 are addressed by the presented replication strategies. An overview of the mapping of requirements and related work is presented in the Tables 3.6 and 3.7. These tables use the following symbols. First, *black-filled circles* are used in case a requirement is fully supported by a replication strategy. Second, *grey-filled circles* denote requirements that are partly addressed. For example, strategies that target improvement of query efficiency without focussing on user interaction are symbolised with a grey-filled circle. Last, *white-filled circles* are used if a requirement is not addressed.

As expected, there is no single strategy that addresses all of the defined requirements. In order to identify the approaches that are closest to the proposed strategies, the strategies presented in Section 3.1 are filtered and divided into two groups. Group 1, consists of replication strategies that target active optimisation of query efficiency (covered by requirements R3 and R4, see Section 3.2.1). Group 2, captures semi-transparent strategies that provide applications with means for affecting replica organisation (covered by requirement R6, see Section 3.2.2). All approaches assigned to these groups are further analysed with respect to the remaining requirements (R1, R2, R5, R7, R8) as well as the challenges of smart products presented in Section 2.2.

3.2.1 Active Optimisation of Query Efficiency

While there are several strategies that (i) aim at improving query efficiency and (ii) base replication decisions on estimations of upcoming demand, only few strategies com-

Replication Strategy	R1	R2	R3	R4	R5	R6	R7	R8
Peer-to-Peer Storage Systems								
OS Deep Archival Storage [KWZ+00]	○	○	○	○	○	○	●	●
OS Put [KWZ+00]	○	○	○	○	○	○	●	●
OS Introspective Replica Mgmt [KWZ+00]	●	○	◍	○	○	○	●	●
OS Caching [KWZ+00]	●	○	○	◍	○	○	●	●
OS TTL-based Replication [Gee02]	●	○	○	◍	○	○	●	●
Freenet Insert [CSWH01, ATS04]	●	○	○	◍	○	○	●	●
Freenet Request [CSWH01, ATS04]	●	○	○	◍	○	○	●	●
PAST Replication [DR02, RD01b]	○	○	○	○	●	○	○	●
PAST Caching [DR02, RD01b]	●	○	○	◍	○	○	●	●
DRP Local [On04]	●	○	○	◍	○	●	●	●
DRP Highly Up First [On04]	●	◍	○	◍	○	●	●	●
DRP Highly Available First [On04]	●	◍	○	◍	○	●	●	●
DRP Combined [On04]	●	◍	○	◍	○	●	●	●
MAS-x [MN05]	○	○	◍	○	◍	○	●	●
sticky-rep [MN05]	○	○	○	○	◍	○	●	●
sticky-ap [MN05]	○	○	◍	○	◍	○	●	●
Top-K [KRT06]	●	○	○	○	●	○	●	●
Top-K Most Frequently Used [KRT06, KRT07]	●	○	○	○	●	○	●	●
ECR Random [LYC07]	○	○	○	○	○	○	●	●
ECR Group Partition [LYC07]	○	○	○	○	○	○	●	●
ECR Highest Available First [LYC07]	○	○	○	○	○	○	●	●
Dynamo [DHJ+07]	○	○	○	○	◍	○	●	●
IRM Query Initiating Rate [She10]	●	○	○	◍	●	○	●	●
IRM Query Passing Rate [She10]	●	○	○	◍	●	○	●	●
Content Distribution Networks and Web Caching								
MDCDN Direct Demand [AMAM04]	◍	○	◍	◍	●	○	◍	◍
MDCDN Indirect Demand [AMAM04]	◍	○	◍	○	●	○	◍	◍
SRP Highly Available First [On04]	○	○	○	○	○	●	●	○
SRP Transit Node [On04]	○	◍	○	○	○	●	●	○
SRP COM [On04]	○	◍	○	○	○	●	●	○
SRP Admission Control [On04]	○	◍	○	○	○	●	●	○
Latency Guranatees [HA04]	○	○	○	◍	○	●	●	●
CQORP [BB04]	○	○	○	◍	○	○	●	○

Table 3.6: Mapping of Requirements and Related Work (1/2)

Replication Strategy	R1	R2	R3	R4	R5	R6	R7	R8
Grid Computing								
Best Client Replication [RF01]	●	○	○	●	○	○	●	●
Cascading Replication [RF01]	●	●	○	●	○	○	●	●
Fast Spread [RF01]	●	●	○	●	○	○	●	●
Workflow-based Replication [CDL+07]	○	○	●	●	○	○	○	●
Best Reply [YWD10]	●	○	○	●	○	○	●	●
PHFS [KIS11]	●	●	●	●	○	○	●	●
Client / Server Storage Systems								
Worklet-Based Replication [BWYH06]	○	○	●	●	○	○	●	○
Hybrid Replication [DBEN07]	○	○	●	○	●	○	○	●
Piggybacking in Mobile Scenarios [SGAM11]	○	○	●	●	●	○	●	○
Cache Prefetch and Replacement [JIR+09]	○	○	●	●	○	○	●	○

Table 3.7: Mapping of Requirements and Related Work (2/2)

bine these two approaches. Thus, group 1 consists of the two workflow-based replication strategies Workflow-based Replication (see Section 3.1.5.2) and Worklet-based Replication (see Section 3.1.6.1), the piggybacking approach proposed by [SGAM11] (see Section 3.1.6.3), as well as the direct-demand-driven strategy employed by MDCDN (MDCDN Direct Demand, see Section 3.1.4.1). Moreover, even though the strategies PHFS (see Section 3.1.5.4) and Cache Prefetch and Replacement (see Section 3.1.6.4) support both requirements R3 and R4, they do not seem to be applicable in the domain of smart products. On the one hand, PHFS is tailored to hierarchical system structures that cannot be generally assumed for smart product systems. On the other hand, Cache Prefetch and Replacement relies on a central deployment, which is not in line with the distributed storage concept of smart products presented in Section 2.2.2.2.

Workflow-based Replication [CDL$^+$07]. The strategy Workflow-based Replication utilises knowledge of upcoming workflow-related content needs and pre-replicates the latter for improving query efficiency. However, the approach assumes non-branched workflow structures, i.e., explicit knowledge of content that will be used during workflow execution. Consequently, the strategy does not cope with uncertainty in the control flow of workflows and the risk of false pre-replication described in Section 2.2.2.1.

Worklet-based Replication [BWYH06]. Worklet-based Replication also makes use of workflows, user profiles, and environmental information to predict future demand and pre-replicate content, accordingly. Even though the authors describe a central operation of the strategy, the concept appears to be realisable in a distributed architecture as well.

However, there is no formal description of the strategy. This strongly limits its assessment with regard to the challenges of smart products.

Piggybacking in Mobile Scenarios [SGAM11]. The combined SPA and piggybacking concept nicely supports the resource limitation of smart products by reducing pre-replication to the next predicted request. The approach is used in a central infrastructure; yet the operation in a distributed mode appears to be feasible. However, the strategy limits its prediction on access patterns, only, and does not take into account problem-solving knowledge of smart products. Also, the limitation to the next expected request may not take advantage of the full potential of pre-replication, even though it clearly limits pre-replication wastage. This is also phrased by the authors, who state that "integrating task and context knowledge can improve these results [prediction accuracy]" [HS07, p. 6].

MDCDN Direct Demand [AMAM04]. MDCDN Direct Demand employs a statistic demand forecasting method to enable content pre-replication. Even though this is comparable with the afore-described SPA and piggybacking concept, it stronger relies on temporal locality properties of access patterns, which is stated to be observable in smart product systems (see Section 2.2.2.2). Also, the approach has a longer pre-replication reach and can be assumed to better balance pre-replication wastage with improvement of query efficiency. Hence, even though MDCDN Direct Demand does not utilise problem-solving knowledge of smart products, it appears to be a promising approach for optimising query efficiency and content availability in smart product systems.

3.2.2 Application-Affected Replica Placement

The second group of replication strategies consists of the strategy Latency Guarantees proposed by [HA04] (see Section 3.1.4.3) as well as the concepts of [On04] presented in Sections 3.1.3.5 and 3.1.4.2. Note that active strategies that employ implicit or explicit application hints are not captured by this group, because they typically focus on upcoming requests, only, and do not provide any means for classifying content and affecting replica organisation.

Latency Guarantees and the P2P replication strategies presented in Section 3.1.3.5 are demonstrated to optimise query efficiency and content availability, respectively. However, they are both limited to a single quality attribute and do not support a generic content classification and a corresponding adaptation of number and placement of replicas.

The static replication strategies for CDNs presented in Section 3.1.4.2 are not applicable in the domain of smart products, because of their assumption of global knowledge about system state and access patterns. Hence, none of the replication strategies

enabling applications to affect replica organisation fully supports the requirements presented in Section 2.4.

3.2.3 Conclusion

Overall, while the workflow-based strategies do not support control flow uncertainty that is stated as a major challenge in smart product systems (see Section 2.2.2.1), the combined SPA and piggybacking concept as well as MDCDN do not leverage procedural knowledge to enhance content organisation. The application-affected replication strategies summarised in Section 3.2.2 are limited to a single objective to adapt replica organisation for and do not enable generic content classification. Hence, in conclusion, none of the approaches fully meets the requirements and challenges of smart products stated in Sections 2.4 and 2.2.1, respectively.

3.3 Chapter Summary

This chapter presented state-of-the-art content placement strategies of the following areas: P2P storage systems, CDNs, Grid computing, and client / server-based storage systems. Each strategy was described and categorised using a novel, overarching classification scheme. Also, the extent to which the strategies support the requirements investigated in Section 2.4 was analysed.

Since none of the replication strategies addresses all of the defined requirements, they were grouped into approaches that (i) target active optimisation of query efficiency and (ii) provide applications with simple means to enable semi-transparent replica organisation. The strategies belonging to these groups were assessed according to the requirements as well as the challenges of smart products.

The analysis revealed that even after grouping the replication strategies, none of the them fits the specifics of smart products. On the one hand, the active replication strategies that leverage procedural knowledge assume simple workflow structures and abstract from uncertainty in the control flow of workflow. Yet, as described in Section 2.2.2.1, this is a common challenge seen in existing smart product application scenarios and is expected to gain importance in the future, in particular in complex manufacturing and maintenance scenarios. On the other hand, there is no replication strategy that provides applications with a generic content classification concept used to enhance content organisation.

The novel content placement strategies developed in this thesis try to fill the aforementioned gaps. They are explained in detail in the following chapter and are categorised using the proposed classification scheme.

Chapter 4
Content Placement Strategies for Smart Products

This chapter presents six novel content placement strategies for smart products that address the overall goals presented in Section 1.2 as well as the requirements for content placement strategies for smart products explained in Section 2.4. Given that many operations of smart products are based on well-defined workflow models, this thesis proposes three active replication strategies: Most Probable Path (MPP), Path Assessment (PA), and Cooperative Path Assessment (CPA). These replication strategies utilise workflow structures to predict and pre-replicate upcoming workflow-related content needs in order to enhance query efficiency of associated content requests as well as overall workflow execution performance. Given the structural complexity of workflows explained in Section 2.2.2.1, the strategies address the questions of (i) when to process pre-replication, (ii) for how many activities to pre-replicate in advance, (iii) how to handle uncertainties in the control flow of workflows, and (iv) how to deal with falsely pre-replicated content. While MPP, PA, and CPA pursue the same objective, they vary in the way of balancing enhancement of query efficiency with the potential risk of false pre-replication as well as in their incorporated sphere of knowledge. Note that the proposed workflow-based replication strategies address local workflow execution, only. A proposal on how the strategies could be used / adapted to support distributed workflows is presented in [MSB11].

The workflow-based replication strategies primarily target query efficiency of workflow-related content requests. In order to optimise content availability and query efficiency independent of workflow-related access patterns, the workflow-based replication strategies are complemented by the replication strategy Content Class (CC) developed in this thesis. CC applies the concept of content classes assigned by applications to express the purpose of the content they are operating with. This way, CC provides case-by-case strategies to adjust number and placement of replicas depending on the application-driven content classification.

Finally, the two content replacement strategies MFR for Transient Replicas (MFRTR) and Enhanced Content Replacement (ECR) are proposed. Both approaches are aligned

with the defined replication strategies and complete the set of content placement strategies for smart products. While MFRTR addresses replicas pre-replicated by the workflow-based replication strategies, ECR is content-class-driven and complements the replication strategy CC.

This chapter is structured as follows. First, Section 4.1 presents the three workflow-based replication strategies MPP (see Section 4.1.2), PA (see Section 4.1.3), and CPA (see Section 4.1.4). Second, Section 4.2 describes both the concept of content classes and its application by the proposed replication strategy CC. Thereafter, the novel replacement strategies MFRTR and ECR as well as a discussion of deployment and operation options of the proposed content placement strategies in the domain of smart products are presented in Sections 4.3 and 4.4, respectively. The extent to which the proposed approaches support the defined requirements is analysed in Section 4.5. Section 4.6 concludes the chapter with a short summary.

This chapter is based upon and partly reuses descriptions out of the following publications of the author of this thesis: [MSH09, MSB11]. Note that paragraphs reused from these publications are not marked as direct quotes if they were written exclusively by the author of this thesis. Quotations within reused paragraphs are taken over and marked accordingly.

4.1 Workflow-based Replication Strategies for Smart Products

4.1.1 Motivation

Traditional workflow engines typically couple and synchronously process the control flow and the data flow of a workflow. This results in on-demand location and retrieval of activity-related content needs as depicted in Figure 4.9(a). Depending on the amount of content required during activity processing, the location of this content, and the network condition, this might result in significant delays in the control flow, impacting overall workflow execution performance.

This thesis proposes the three workflow-based replication strategies MPP (see Section 4.1.2), PA (see Section 4.1.3), and CPA (see Section 4.1.4) that decouple the control flow and the data flow of a workflow as proposed by [CDL+07, DC08]. They assume activities to be annotated with content needs and utilise workflow structures to predict and pre-replicate future activity-related content needs. This enhances query

efficiency of content requested in the control flow and – hence – overall workflow execution performance.[32]

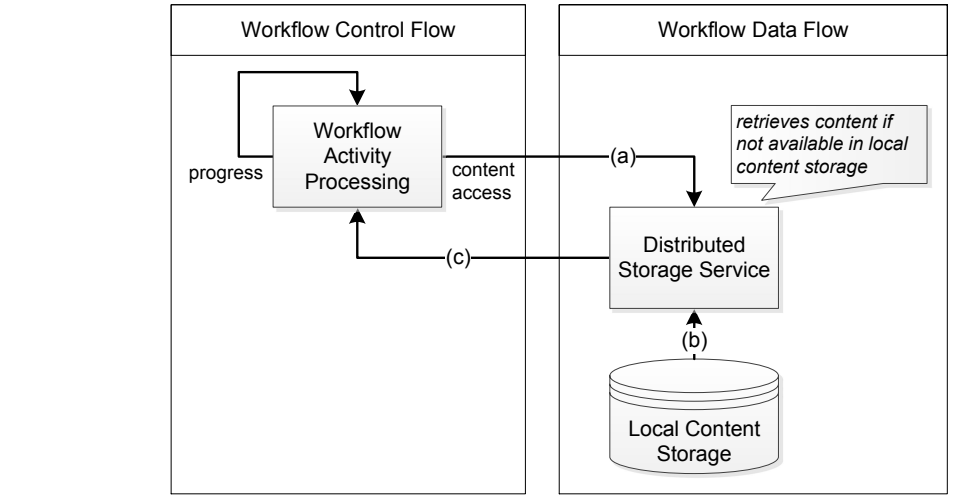
Yet, conditioned XOR transitions and resulting branches in the control flow may lead to uncertainty in workflow execution and implies the risk of pre-replication wastage (see Section 2.2.2.1). Each of the proposed replication strategies balances this inherent trade-off between enhancement of query efficiency of activity-related content demand and the risk of false pre-replication to a different extent. Especially in scenarios with resource-limited smart products, proper balancing of this trade-off as well as the message and computation complexity of replication strategies are both of high importance and may affect the suitability of the latter. In general, it can be stated that the lower the resources possessed by smart products, the more restrictive a replication strategy should be in terms of pre-replication wastage and complexity. This aspect is addressed by the three proposed workflow-based replication strategies and is analysed in detail in the evaluation study presented in Chapter 5.

Common Phases. The strategies MPP, PA, and CPA all consists of three consecutive phases. An overview of these phases including the resulting interrelation between the control flow and the data flow of a workflow is presented in Figure 4.9(b).

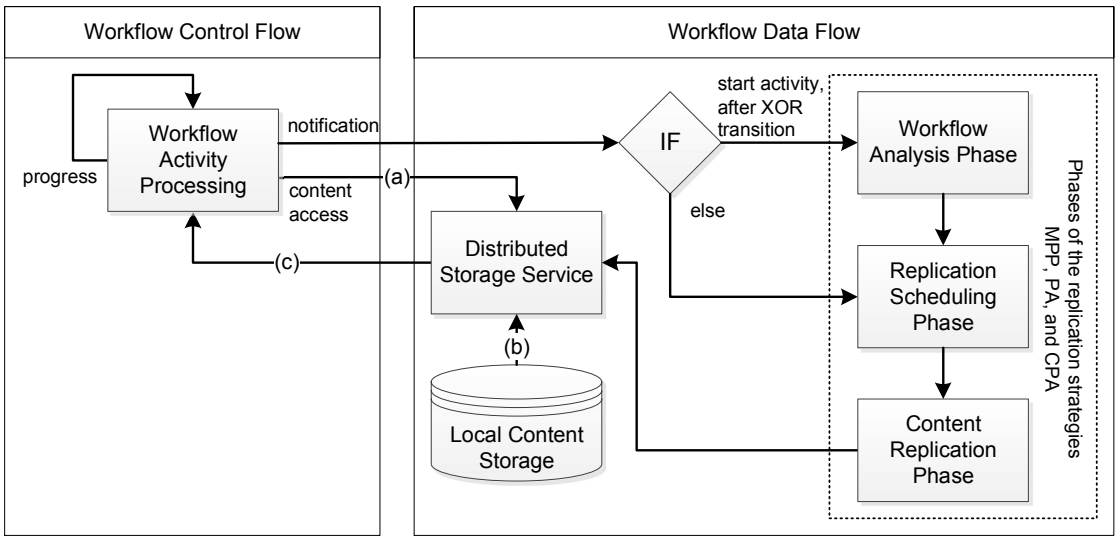
1. *Workflow Analysis Phase:* In this phase, all alternative paths of a given workflow are determined and assessed based on a dedicated metric. The phase is processed by a smart product starting the execution of a workflow as well as after processing related XOR transitions.
2. *Replication Scheduling Phase:* All paths with assessment results meeting strategy-specific selection criteria are further processed in the second phase, in which content retrieval is scheduled. This schedule is regularly adjusted according to the progress in the control flow in order to reduce the risk of false pre-replication.
3. *Content Replication Phase:* In this phase, the schedule is processed and – so called – *transient replicas* (see Section 4.1.2.3) are created to decrease access latency and increase workflow execution performance.

MPP. MPP is the most restrictive of the proposed replication strategies. It assesses paths according to their execution probability and pre-replicates activity-related content needs of the path with the highest execution probability, only. While this reduces the risk

[32] Note that there is no consistent use of the term "coupling" in literature related to workflow operations. For example, [BC09] makes use of the term "coupling" to describe the interaction degree between the workflow engine and the distributed storage service covering the control flow and the data flow, respectively. This way, [BC09] distinguishes three coupling degrees. First, *decoupled strategies* receive content needs of all activities prior to workflow execution. Second, *loosely-coupled strategies* apply on-demand retrieval of activity-related content needs, i.e., for each activity content needs are retrieved when the activity is being processed. This equals the interrelation depicted in Figure 4.9(a). Finally, tightly-coupled strategies enable enhanced retrieval of activity-related content needs based on regular exchange between the workflow engine and the distributed storage service. This case matches the concept of the proposed replication strategies depicted in Figure 4.9(b).



(a)



(b)

Figure 4.9: Overview of the Interrelation of the Control Flow and the Data Flow of a Workflow: (a) Coupled Control Flow and Data Flow with On-Demand Content Retrieval, (b) Decoupled Control Flow and Data Flow with Proactive Content Retrieval Using the Proposed Replication Strategies MPP, PA, and CPA

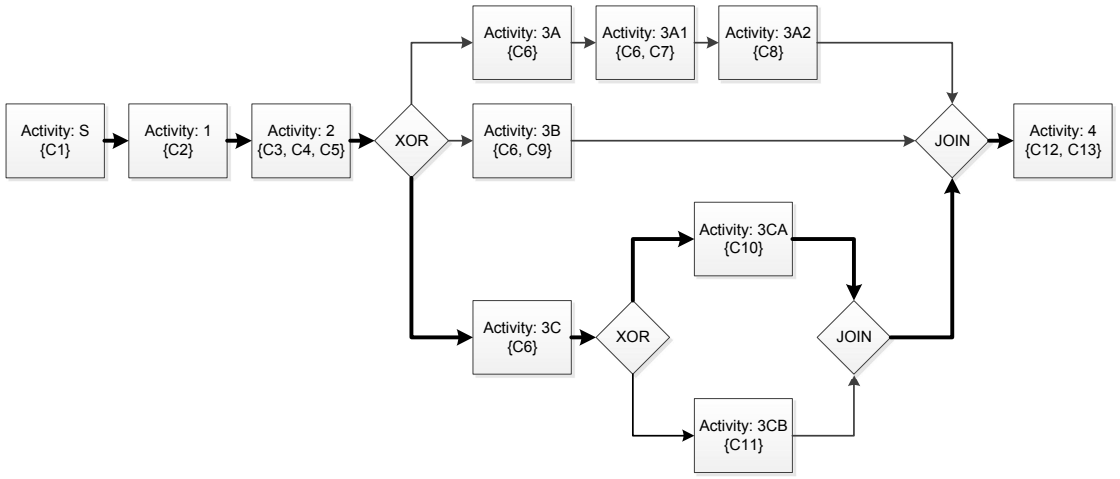


Figure 4.10: Workflow Example

of false pre-replication, it may temporarily fall back to on-demand content retrieval in case another but the most probable path is being followed.

PA. Compared to MPP, PA consists of a more sophisticated path assessment metric that combines path execution probabilities with measurements of replication cost and benefit for each content needed by potentially upcoming activities. This approach may lead to more than one path being processed in the replication scheduling phase if the corresponding path assessment value exceeds a pre-defined threshold. For this reason, PA incorporates an adapted scheduling mechanism that addresses the increased pre-replication wastage. The content replication phase is equal to the approach used by MPP.

CPA. While both MPP and PA represent local replication strategies, CPA adapts PA by incorporating an extended sphere of knowledge in the workflow analysis phase. In addition to the content interest of the smart product executing the workflow, the mechanism further takes into consideration content interests of surrounding smart products. This is based on the assumption that smart products nearby the one executing the workflow might query the latter for workflow-related content and benefit from improved query efficiency. CPA infers knowledge about geographic locality properties of content access patterns of smart products belonging to the same environment, which is utilised to enable a more accurate replication cost / benefit calculation. The replication scheduling phase and the content replication phase equal the approach used by PA and MPP, respectively.

Workflow Example. In the following sections, the workflow depicted in Figure 4.10 is used to exemplify the functionality of the proposed strategies. This workflow consists of 3 sequential activities (S, 1, 2) followed by an XOR transition. This transition

splits the workflow in 3 alternative paths that all consist of the common sub path (S, 1, 2) but differ from activity 2 onward. Moreover, the third path (S, 1, 2, 3C) is again split in 2 alternative paths after activity 3C by means of another XOR transition. Hence, the workflow consists of a total number of 4 alternative paths that are structured as follows: $p_1 = (S, 1, 2, 3A, 3A1, 3A2, 4)$, $p_2 = (S, 1, 2, 3B, 4)$, $p_3 = (S, 1, 2, 3C, 3CA, 4)$, $p_4 = (S, 1, 2, 3C, 3CB, 4)$. The number of content objects needed by an activity is denoted below the activity name in curly brackets. Moreover, the path p_3 highlighted with bold arrows is assumed to have the highest execution probability. Finally, for reasons of simplicity, all activity-related content needs are assumed to be of the same size (this applies to the example, only, and is not an assumption made by the proposed replication strategies).

Workflow Transitions. Note that neither the above-presented workflow nor the three strategies presented in the following sections address transitions of type AND or OR directly. AND transitions do not result in alternative but parallel paths. Given the assumption that synchronisation issues are covered in the control flow, the proposed replication strategies can handle parallel paths as a single path that combines content needs of parallel activities.

As opposed to XOR transitions, OR transitions may lead to multiple of the alternative paths being followed in parallel. For example, using the workflow depicted in Figure 4.10, the paths p_1 and p_2 could both be processed if the transition was an OR transition. While this scenario is out of scope of MPP, which solely pre-replicates activity-related content needs of the most probable path, it is covered by PA and CPA that may both pre-replicate content needs of activities belonging to different paths. Yet, after passing an OR transition in the control flow, the parallel paths can again be handled as if they were based on an AND transition.

4.1.2 Replication Strategy: Most Probable Path (MPP)

4.1.2.1 Workflow Analysis Phase

Path Determination. In the first step of the workflow analysis phase, all alternative paths of the workflow are determined, starting from the activity being processed at this time. As stated above, this activity can either be the start activity of the workflow or one of the activities following the last processed XOR transition. Since the determined paths provide the basis for the following phases, their length is limited in order to reduce uncertainty and the risk of pre-replication wastage. Thus, path length is limited in case of XOR transitions, after which a maximum number of α activities are considered. Moreover, in case of successive XOR transitions, α is reduced to only include activities up to the second XOR transition (i.e., only a single XOR transition is covered). Given the

$$f^{MPP}(p,T) = \sum_{t=1}^{T} \frac{1}{T-t+1} \times \frac{h_t^p}{h_t} \quad (1)$$

p : path to be evaluated, $p \in P$

T : current period

h_t^p : number of complete executions of path p in period t

h_t : number of complete path executions in period t

example presented in Fig. 4.10 and α being configured with a value of 2, this results in the set $P = \{p_1 = \{S, 1, 2, 3A, 3A1\},\ p_2 = \{S, 1, 2, 3B, 4\},\ p_3 = \{S, 1, 2, 3C\}\}$.

Path Assessment. Thereafter, all paths $p \in P$ are assessed based on the execution probability metric $f^{MPP}(p,T)$ presented in Eq. 1. This metric calculates the path execution probability by the ratio between the number of completed executions h_t^p of a path p in period t and the overall number of path executions h_t of the given workflow in period t. Moreover, the metric applies the concept of temporal locality and weights path execution by the period t in which it took place with recent executions having a stronger impact. Thus, path executions that occurred in the current period T are assigned a weighting factor of 1. Path executions that occurred in previous periods t^* are weighted by the multiplicative inverse of the number of periods that passed since the path executions took place (t^* and T included).

The effect of the weighting factor can be controlled by the configuration of the period length. Configurations with a short period length intensify the impact of recent path executions. In contrast, the effect of path execution recency can be weakened in case of a sufficiently long period length setting. An exemplary configuration of the period length is presented in the evaluation study in Table 5.20 (see parameter $t_{WORKFLOW}$).

As an example, imagine path p_1 of the above workflow being processed once in period $t = 1$, two times in period $t = 3$, and the current period being configured as $T = 4$. Moreover, imagine the workflow being processed two times per period. In this example, the execution probability of path p_1 is calculated as $f^{MPP}(p_1,T) = \frac{1}{4} \times \frac{1}{2} + \frac{1}{3} \times \frac{0}{2} + \frac{1}{2} \times \frac{2}{2} + \frac{1}{1} \times \frac{0}{2} = \frac{5}{8}$.

Result Set. MPP selects the path p^* with the highest execution probability $f^{MPP}(p^*,T)$ for further processing in the subsequent phases. Note that in case of multiple paths with identical execution probabilities, the strategy selects the path that was processed most recently.

4.1.2.2 Replication Scheduling Phase

In the replication scheduling phase, a schedule is defined for retrieving content that is needed by the upcoming activities of p^* and not stored on-board of the smart product executing the workflow.

Schedule Reach. Given the limited storage capacity of smart products, misconfiguration of the schedule reach could lead to the risk of "overwriting" pre-replicated content. This means, content being pre-replicated for one of the upcoming activities could be replaced / removed by content being pre-replicated for activities succeeding the latter. Imagine a smart product that executes the workflow depicted in Figure 4.10 and that is only capable of storing an additional set of 5 content objects (for simplicity, all content objects are assumed to be of the same size in this example). Moreover, imagine that MPP starts pre-replication of content $C6$ related to activity $3C$ while activity S is being processed in the control flow of the workflow. Depending on the applied replacement strategy, this could result in the content object $C2$ being replaced for $C6$. This would strongly affect query efficiency, because of the "wrong" content being made available given the state of the control flow.

Misconfiguration of the schedule reach moreover increases the risk of false pre-replication. For example, the impact of an aborted workflow execution (e.g., because of a product failure or a user-triggered abortion of workflow execution) on the amount of falsely pre-replicated content can be reduced by limiting the length of the schedule.

Iterative Scheduling. To reduce conflicting pre-replication given the resources of smart products as well as to limit the risk of false pre-replication, the schedule of the replication strategy MPP is *linked with control flow progress*. In a first step, all activities of the path p^* are organised in a queue in the order of their expected execution in the control flow of the workflow. This queue is processed iteratively by the scheduler, which – consequently – only covers a subset of the queue at any time. The actual number of activities within this subset, i.e., the actual schedule reach, is managed by a configurable parameter τ. Being initialised with a number of τ activities, the schedule is extended by another $\lceil \frac{\tau}{2} \rceil$ activities after a progress of a number of $\lceil \frac{\tau}{2} \rceil$ activities in the control flow. This results in a schedule consisting of content needs related to a maximum number of τ activities in all iterations.

As an example, image a path p consisting of 6 activities $p = \{A1, A2, A3, A4, A5, A6\}$ and τ being set to a value of 3. This leads to the following schedules S_i with i referring to the progress in the control flow of the workflow, i.e., the activity being processed: $S_{A1} = \{A1, A2, A3\}$, $S_{A2} = \{A2, A3\}$, $S_{A3} = \{A3, A4, A5\}$, $S_{A4} = \{A4, A5\}$, $S_{A5} = \{A5, A6\}$, $S_{A6} = \{A6\}$. Note that this example does not cover activity removal from the schedule because of all related content needs having been retrieved, which might further shorten the schedule from head to tail.

Scheduling & XOR Transitions. In addition to the progress of the control flow, rescheduling is managed based on the workflow structure. Thus, the schedule is only extended by a number of $\lceil \frac{\tau}{2} \rceil$ activities, if the activity being processed (i.e., the one triggering the extension as described afore) is *not followed by an XOR transition*. This is reasonable, because the schedule is cleared and built from scratch after processing XOR transitions (see Figure 4.9(b)). For example, given the workflow presented in Figure 4.10 with the most probable path $p^* = p_3$, τ being set to a value of 4, and the assumption of p_3 being the actual path processed in the control flow, this leads to the following schedule: $S_S = \{S, 1, 2, 3C\}$, $S_1 = \{1, 2, 3C\}$, $S_2 = \{2, 3C\}$, $S_{3C} = \{3CA, 4\}$, etc. Note that there is no schedule extension made at activity 2 because it is followed by an XOR transition. Instead, the schedule is cleared and built from scratch when processing activity 3C.

While this approach further reduces the risk of false pre-replication, it might temporary fall back to on-demand retrieval of activity-related content needs depending on the configuration of τ. If τ is set to a value of 3 in the above example, then the activity 3C is not covered in the schedule $S_S = \{S, 1, 2\}$. Consequently, content needs of activity 3C have to be retrieved – similar to activity S – on-demand when processing activity 3C (i.e., synchronous with the control flow). This again illustrates the need of balancing the trade-off between query efficiency and pre-replication wastage.

4.1.2.3 Content Replication Phase

In the *content replication phase*, the schedule is processed and content retrieval is performed. Moreover, in order to limit load in terms of network traffic effected by the replication strategy, content retrieval is controlled by a configurable maximum number of parallel requests ζ . An exemplary calculation of the number of parallel requests as a function of τ is presented in Section 5.4.1.

Transient Replicas. Since, in contrast to reactive replication strategies, MPP does not rely on high access rates occurred in past periods, it merely creates *transient replicas*. Transient replicas are temporary available only and are assigned a configurable gradually increasing TTL with pre-defined TTL intervals. For example, these TTL intervals could be configured in terms of time units as $TTL_1 = 100$, $TTL_2 = 200$, $TTL_3 = 400$. Following the concept of temporal locality, the TTL of a transient replica is increased per access in order to extend its availability in the network. For example, if a transient replica with the active TTL interval TTL_2 is requested, its TTL is increased to the next interval TTL_3. If the access rate of a transient replica exceeds a configurable threshold, then it is transformed into a (persistent) replica. In the above example, this transformation would be performed, if a transient replica with an active TTL interval TTL_3 was accessed again.

Lazy Removal. If a transient replica is not accessed during its active TTL interval, it might be either removed or flagged as candidate to be removed depending on the capa-

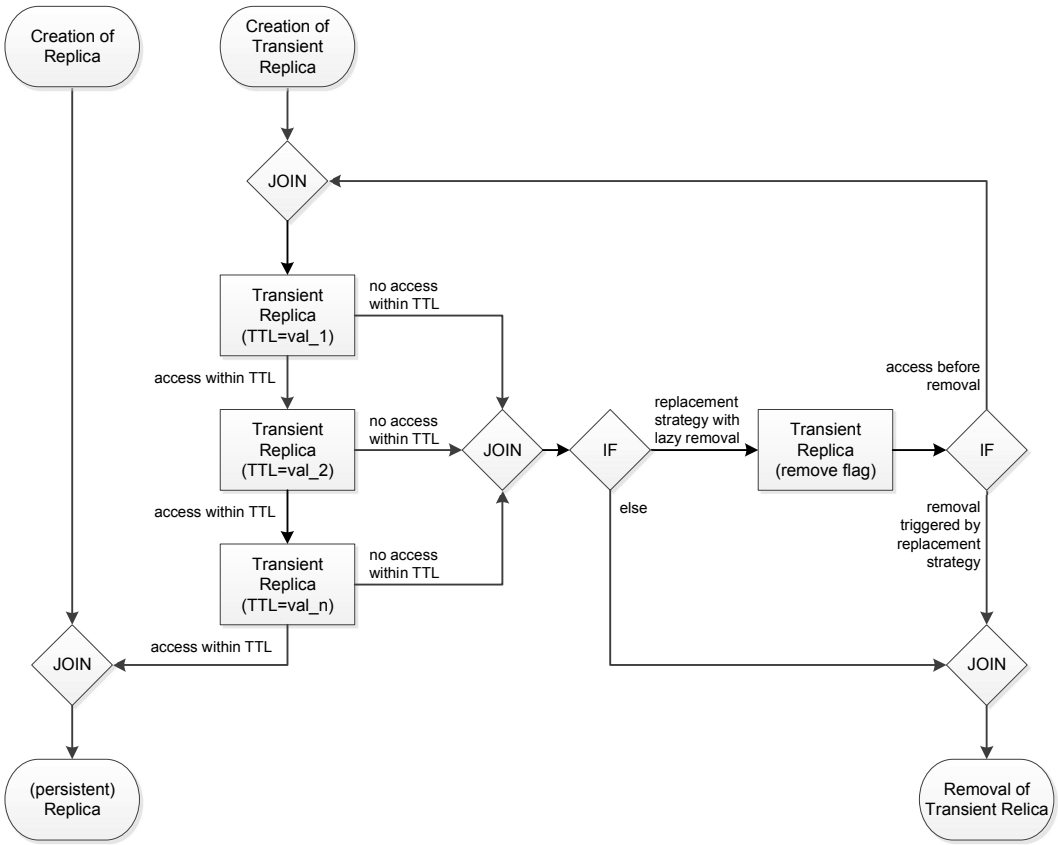


Figure 4.11: Replica Life Cycle

bilities of the applied replacement strategy. As opposed to direct removal, this kind of *lazy removal* enables such candidates to be "re-initiated" in case they are accessed prior to being eventually removed by the replacement strategy. In this case, the remove flag is revoked and the TTL is set to its initial value. Consequently, the concept of transient replicas avoids flooding the network with replicas solely required during workflow execution and reduces related content replacement operations. An overview of the proposed replica life cycle is presented in Figure 4.11. An example of a replacement strategy that incorporates the lazy removal concept of transient replicas is presented in Section 4.3.

Finally, according to the schedule adjustment described afore, transient replicas are created only if the associated activity has not been removed from the schedule in the time between pre-replication request and object retrieval. Otherwise, no transient replica is created and the content is dropped. While this has no impact on network traffic, it reduces the overall replication degree and avoids potentially unnecessary transient replicas.

4.1.2.4 Approximation of Pre-Replication Failure

This section presents an analysis of the potential pre-replication failure of the proposed replication strategy MPP exemplified by the workflow presented in Figure 4.10. The analysis approximates the theoretical pre-replication failure both for the average and the worst case, and facilitates comparison of different configurations. This approach is also applied as part of the evaluation study in order to frame the measured pre-replication failure. Note that there is no separate analysis for the replication strategies PA and CPA, since their multi-path pre-replication concept requires many assumptions or distinction of cases. For this reason, the below-presented approach is used to approximate a lower bound for the pre-replication failure of the two more optimistic strategies PA and CPA.

Assumptions. The analysis assumes execution of alternative paths to be distributed uniformly. Hence, given the workflow presented in Figure 4.10, the alternative activities 3A, 3B, and 3C following the first XOR transition are all processed with the identical probability of 33.33%. Moreover, content size is not taken into account by the analysis, which covers the absolute number of falsely pre-replicated content, only.

Procedure. For each alternative path, the approach calculates the number of potentially falsely pre-replicated content subject to the configuration of τ and α. Remember that these parameters affect schedule length and – consequently – the number of activities for that associated content is replicated in advance. Moreover, the probability for processing certain paths (especially in case of consecutive XOR transitions) as well as the probability that the path for which pre-replication has been conducted is not processed are taken into account. Finally, in order to determine the pre-replication failure rate, the number of falsely pre-replicated content in the average and worst case is set in relation to the average number of content needed by the alternative paths.

Exemplary Application. Figure 4.12 illustrates an example of this analysis for the workflow presented in Figure 4.10. This example assumes three different configurations for τ and α to analyse the effect of schedule reach on pre-replication failure. The number of content needed during workflow execution averages out at a value of 9.5 (content needs per path: $p_1 = 11$, $p_2 = p_3 = p_4 = 9$). First, as mentioned in Section 4.1.2.2, the configuration $(\tau, \alpha) = (3, 1)$ leads to content needs of activity 3C being retrieved on-demand. Hence, while this worsens query efficiency, there is no risk of false pre-replication for the XOR transition following activity 2. The second XOR transition following activity 3C results in two alternative paths that are both assigned an execution probability of 50% given the above assumption. Moreover, only content needed by the activities 3CA and 3CB may be subject to false pre-replication (one content object each); content needed by activity 4 is required in either case. To take into account dependent probabilities, the calculation multiplies the average pre-replication failure at the second XOR transition by the execution probability of the corresponding path $(S, 1, 2, 3C)$, which equals a value of 33.33%. This leads to an average pre-replication failure of $\frac{1}{3} \times \frac{1}{2} \times \frac{1}{2} \times (1 + 1) = 0.17$ content objects. Note that the two factors $\frac{1}{2}$ are required

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
2	0.33	2.00	3CA, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
	0.33	2.00	3CB, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
							0.17

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	3.00	3A	T	0	0.33	0.00
	1.00	3.00		F	1	0.67	0.22
	1.00	3.00	3B	T	0	0.33	0.00
	1.00	3.00		F	2	0.67	0.44
	1.00	3.00	3C	T	0	0.33	0.00
	1.00	3.00		F	1	0.67	0.22
2	0.33	2.00	3CA, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
	0.33	2.00	3CB, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
							1.06

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	3.00	3A, 3A1	T	0	0.33	0.00
	1.00	3.00		F	3	0.67	0.67
	1.00	3.00	3B, 4	T	0	0.33	0.00
	1.00	3.00		F	2	0.67	0.44
	1.00	3.00	3C	T	0	0.33	0.00
	1.00	3.00		F	1	0.67	0.22
2	0.33	2.00	3CA, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
	0.33	2.00	3CB, 4	T	0	0.50	0.00
	0.33	2.00		F	1	0.50	0.08
							1.50

Figure 4.12: Approximation of Pre-Replication Error: Workflow Example

to take into account (i) the probability that a certain path is subject to pre-replication and (ii) the probability that another path is actually followed. Looking at configuration $(\tau, \alpha) = (4, 2)$, these factors are set to a value of (i) $\frac{1}{3}$ and (ii) $\frac{2}{3}$ for the first XOR transition.

From the results depicted in Figure 4.12 it can be derived that the longer the reach of the schedule, the higher the pre-replication failure. Yet, longer schedule reach most likely results in enhanced query efficiency. The configuration with $(\tau, \alpha) = (3, 1)$ that leads to on-demand retrieval of content needs of activities following the first XOR transition can be taken as an example. This again illustrates the trade-off between enhancement of query efficiency and pre-replication failure, which is analysed in detailed in the evaluation study presented in Section 5.4.2.

4.1.3 Replication Strategy: Path Assessment (PA)

The replication strategy PA bases upon MPP. It extends the functionality of the workflow analysis phase and of the replication scheduling phase defined by MPP. The functionality of the content replication phase defined by MPP is reused by PA.

4.1.3.1 Workflow Analysis Phase

Adapted Path Assessment. The replication strategy PA extends the workflow analysis phase of MPP to enable multi-path pre-replication. After determining the set of alternative paths P as described in Section 4.1.2.1, PA applies the extended path assessment metric $f^{PA}(p, T)$ presented in Eq. 2. This metric adopts the concept of weighted path execution probabilities used by MPP. However, PA makes use of the parameters h_t^{pz} and h_t^z that extend the parameters h_t^p and h_t, respectively, used by MPP. The parameter h_t^{pz} equals h_t^p if the path p was processed in period t at least once (i.e., if $h_t^p > 0$). Else, if the path p was not executed in period t, then h_t^{pz} is set to a value of $\frac{1}{|Z_t|}$ with the set Z_t consisting of all paths $p \in P$ that were not processed in period t. Consequently, each path that was not processed in a given period is assigned an equally distributed execution probability. To reflect this "virtual" execution probability in the parameter h_t^z and to ensure an overall probability of 100%, the latter is set to a value of $h_t + 1$ in case there is at least one path $p \in P$ that was not processed in period t. Otherwise, h_t^z is equal to the parameter h_t used by MPP.

As an example, imagine the set of paths $P' = \{p_1, p_2, p_3\}$ of the workflow shown in Fig. 4.10 with the following executions in a given period t, $\{h_t^{p_1} = 3, h_t^{p_2} = 0, h_t^{p_3} = 0\}$. According to the extended ratio, the paths would be assigned execution probabilities $\{p_1 = \frac{3}{4}, p_2 = p_3 = \frac{1}{8}\}$. Unlike MPP, which solely relies on path execution history,

this avoids path execution from being a criterion of exclusion and enables an extended assessment of all paths even if they were not executed so far.

The weighted path execution probability is multiplied with the arithmetic mean of pre-replication benefit / cost ratios of activity-related content needs summed up over all activities of the given path. Calculating the arithmetic mean is required, because otherwise activities with a high number of content needs could be assigned higher assessment values than activities with few content needs, even in case of the latter having higher individual content-related pre-replication benefit / cost ratios.

Benefit Calculation. The metric $b^{PA}(d, T)$ for assessing the pre-replication benefit of an activity-related content d given the current period T is presented in Eq. 3. It incorporates the number of requests r_t^d for content d in a period t and weights r_t^d by the period t to account for temporal locality properties. This request-rate-based calculation is frequently applied by reactive replication strategies (see Section 2.1.2). The weighting factor adopts the approach used for the calculation of weighted path execution probabilities.

The resulting weighted request rate is added up to the total number of times the content d is needed in the different paths $p \in P$. This is based on the assumption that the benefit of having a local replica increases with the number of times the content is needed in alternative paths. For example, given the set $P^{'} = \{p_1, p_2, p_3\}$ of the workflow shown in Figure 4.10 with all activities 3A, 3B, and 3C following the first XOR transition needing content $C6$, the creation of a transient replica of $C6$ would be of high benefit regardless of the actual path being processed. Looking at the benefit calculation of path p_1, the value of creating a transient replica of $C6$ further increases because it is also needed for processing activity 3A1. Note that this approach is the reason for D_a^p in Eq. 2 only covering the initial occurrence of a content object d for each path.

Finally, the resulting sum is multiplied with a location factor l_d, which assigns higher benefit to content stored on-board compared to content being stored off-board. This is reasonable taking into consideration that the assessment covers the overall path. Paths are more attractive candidates for pre-replication if a subset of their content needs is already stored on-board of the smart product executing the workflow.

Cost Calculation. The cost of creating a local replica $c^{PA}(d)$ is calculated based on the ratio between content size s_d and the position w_{pa}^d of the first activity in the path p that requires content d (see Eq. 4). Given the limited storage capacity of smart products, the cost value increases with increasing content size. Moreover, according to the iterative scheduling presented afore, weighting content size with regard to the position of the request is based on the assumption that the risk of false pre-replication decreases the later the content is needed in the control flow.

Note that content size is taken into consideration by the cost metric, only. This is because of both the benefit and the cost metric covering the same content d. Moreover, content size is assumed to be constant over time (remember the assumption of

$$f^{PA}(p,T) = \sum_{t=1}^{T}\left(\frac{1}{T-t+1} \times \frac{h_t^{pz}}{h_t^z}\right) \times \sum_{a \in A_p}\left(\frac{1}{|D_a^p|} \times \sum_{d \in D_a^p} \frac{b(d,T)}{c(d)}\right) \tag{2}$$

$$b^{PA}(d,T) = l_d \times \left(\sum_{t=1}^{T} \frac{r_t^d}{T-t+1} + \sum_{p \in P} \sum_{a \in A_p} d_a^p\right) \tag{3}$$

$$c^{PA}(d) = \frac{s_d}{w_{pa}^d} \tag{4}$$

$$h_t^z : \begin{cases} h_t & \text{if } |Z_t| = 0, \text{ with } Z_t := \{p \in P \mid h_t^p = 0\} \\ h_t + 1 & \text{if } |Z_t| > 0, \text{ with } Z_t := \{p \in P \mid h_t^p = 0\} \end{cases}$$

$$h_t^{pz} : \begin{cases} h_t^p & \text{if } h_t^p > 0 \\ \frac{1}{|Z_t|} & \text{if } h_t^p = 0, \text{ with } Z_t := \{p \in P \mid h_t^p = 0\} \end{cases}$$

A_p : set of activities a in path p

D_a^p : set of content objects needed by activity a that are not needed by previous activities of path p

$$d_a^p : \begin{cases} 0 & \text{if } d \notin D_a \\ 1 & \text{if } d \in D_a \end{cases}$$

$$l_d : \begin{cases} \lambda & \text{weighting factor if } d \text{ is stored off-board} \\ \sigma & \text{weighting factor if } d \text{ is stored on-board, with } \sigma > \lambda \end{cases}$$

r_t^d : number of requests for content object d in period t

s_d : size of content d

w_{pa}^d : position of activity a in path p with first *GET* request for content d

immutable content; of course, different content may be of different size). For this reason, including the parameter in the benefit calculation would be equal to adapting the weighting of the content size in the cost metric (e.g., using s_d^2 or $\sqrt{s_d}$).

Result Set. The set of paths to be further processed R^{PA} includes the path p^* with the highest execution probability as well as all paths with assessment values above a configurable threshold γ^{PA} (see Eq. 5). An exemplary configuration of the threshold γ^{PA} is presented in Section 5.4.1.2.

$$R^{PA} := \{p \in P \mid (f^{PA}(p,T) \geq \gamma^{PA})\} \cup \{p^*\} \tag{5}$$

4.1.3.2 Replication Scheduling Phase

Based on the set R^{PA}, an enhanced *multi-path queueing* mechanism is defined. First, similar to the queueing approach used by MPP, the content needs of activities that belong to the sub-path common for all alternative paths $p \in R^{PA}$, i.e., all activities up to the first XOR transition, are put into a queue. Thereafter, activities of the alternative paths are queued in breadth-first order with the path p^* having the highest execution probability being added first and all other paths being added in descending order of their path assessment value. This ensures that the more valuable the content the earlier it is queued for pre-replication.

This way, the queue is iteratively extended with content needs of an overall number of α activities of path p^*. For all other paths $p \in R^{PA} \setminus \{p^*\}$, content needs of an overall number of β activities are queued. The parameter β is configured to a value of $\beta \leq |R^{PA} \setminus \{p^*\}| \times \alpha$ to ensure that for each path $p \in R^{PA}$ the number of activities for which content needs are queued does not exceed the limit of α. The resulting queue of activities is processed by the iterative scheduling concept of MPP described in Section 4.1.2.2.

Since α is used in the workflow analysis phase for path length limitation (see Section 4.1.2.1), this approach does not put any restrictions on the number of activities that can be queued for the paths $p \in R^{PA}$. While all activities of the path p^* can be queued, the number of activities that can be queued for each of the remaining paths $p \in R^{PA} \setminus \{p^*\}$ depends on the configuration of the parameter β. If β is set to its maximum value, all activities of the paths $p \in R^{PA} \setminus \{p^*\}$ can be added to the queue (an exemplary configuration is discussed in see Section 5.4.1.2).

As an example, imagine the set of path $R^{PA} = \{p_1, p_2, p_3\}$ of the workflow depicted in Figure 4.10 with $p_1 = p^*$ (as opposed to the illustration), $f^{PA}(p_2, T) > f^{PA}(p_3, T)$, and $\alpha = \beta = 2$. This would result in the queue $Q = \{S, 1, 2, 3A, 3B, 3C, 3A1\}$.

4.1.4 Replication Strategy: Cooperative Path Assessment (CPA)

The replication strategy CPA bases upon PA and extends the functionality of the workflow analysis phase defined by the latter. CPA makes use of the functionality of the replication scheduling phase and of the content replication phase defined by PA and MPP, respectively.

$$f^{CPA}(p,T) = \sum_{t=1}^{T}\left(\frac{1}{T-t+1} \times \frac{h_t^{pz}}{h_t^z}\right) \times \sum_{a \in A_p}\left(\frac{1}{|D_a^p|} \times \sum_{d \in D_a^p} \frac{b^{CPA}(d,T)}{c^{PA}(d)}\right) \quad (6)$$

$$b^{CPA}(d,T) = l_d \times \left(\sum_{t=1}^{T} \frac{r_t^d}{T-t+1} + \sum_{p \in P}\sum_{a \in A_p} d_a^p\right) + l_d^N \times \left(\sum_{n \in N}\sum_{t=1}^{T} \frac{r_t^{dn}}{T-t+1}\right) \quad (7)$$

N : set of nodes that send index to workflow-processing node

l_d^N : weighting factor for potential content requests from nodes $n \in N$

r_t^{dn} : number of requests from node n for content object d in period t for which n is not a provider

4.1.4.1 Workflow Analysis Phase

While both MPP and PA are classified as local replication strategies, CPA makes use of an extended sphere of knowledge. It incorporates content interest of surrounding smart products in order to infer knowledge about geographic locality properties of access patterns and to enable an enhanced path assessment. This assessment takes into account pre-replication benefit and cost not only from the perspective of the local smart product but moreover from the perspective of its surroundings. While being more complex in terms of knowledge gathering, this approach facilitates making more informed and accurate pre-replication decisions as well as higher utilisation of transient replicas generated during the content replication phase.

Prerequisites. CPA assumes a distributed storage service that routes content requests that cannot be served locally to smart products in the close environment (i.e., neighbours). Moreover, to capture content interest of surrounding smart products, CPA requires each smart product to periodically send an index consisting of direct and indirect access rates (see definition in Section 2.3.2) for all content that the product *is not a provider of* to its best k neighbours. This way, each smart product receives 0..N indices with content interests of its surroundings in regular intervals.[33] Based on the assumption of temporal locality, this enables the receiving smart product to infer / predict content requests it will likely receive in the upcoming period (i.e., the higher the request rate aggregated over all 0..N indices, the higher the probability of receiving another request).

[33] Note that it would not be sufficient to let each node individually maintain indirect requests rates, because this would not reveal whether any of the requesting nodes have become a provider of the requested content.

Adapted Benefit Calculation. While CPA adopts the metrics $f^{PA}(p,T)$ and $c^{PA}(d)$ of PA for the overall path assessment metric and cost metric, respectively, it includes the enhanced benefit metric $b^{CPA}(d,T)$ presented in Eq. 7. This metric extends $b^{PA}(d,T)$ with a second summand to assess the value of the content d for other smart products $n \in N$ given the indices described afore. For each of these smart products, the request rate maintained in the related index is weighted by the period in which requests took place. The resulting values are summed up over all nodes $n \in N$ and multiplied with the configurable location factor l_d^N.

Result Set. Finally, equal to PA, all paths with assessment values above the threshold γ^{CPA} plus the most probable path p^* are further processed (see Eq. 8).

$$R^{CPA} := \{p \in P \mid (f^{CPA}(p,T) \geq \gamma^{CPA})\} \cup \{p^*\} \tag{8}$$

4.2 Content-Class-based Replication Strategies for Smart Products

4.2.1 Content Classes

As described in Section 2.2.1, smart products pursue or serve a well-defined purpose at a given point in time and can be assumed to "know" the use of the content they are operating with. This purpose-driven behaviour provides the basis for the proposed concept of *content classes*. Content classes enable categorisation of content and are assigned by applications of smart products during runtime. This categorisation may affect replica management in terms of number, placement, and type of replicas (see Section 4.2.2). Moreover, locality properties inferred from content classes may be used to enhance replacement operations (see Section 4.3.2). Hence, *applications directly affect content organisation* by means of explicit content class assignment.

Prerequisites. While content placement strategies may leverage content class information, the organisation and maintenance of content class assignments has to be supported and handled by the distributed storage service. Thus, distributed storage services should allow both initiators and requesters for specifying operation-related content classes in order to enable content placement strategies to handle both storage and retrieval activities. Moreover, content may be shared by multiple applications in smart product systems and used for different purposes. Consequently, different applications may associate content with different content classes. This multi assignment of content

classes should be supported by distributed storage services. For example, content class assignments could be maintained as tuples (application identifier, content class) as part of content-related metadata.

In order to facilitate utilisation of content classes also in the case of non-application-triggered operations (e.g., periodic maintenance of replica placement), distributed storage services should be able to determine the *primary content class* of a given content. If there are content class assignments of applications associated with the smart product triggering the operation, then the class most often assigned by these applications is used as primary content class. Else, the primary content class equals the class most often assigned taking into consideration all assignments related to the given content. In the rare case of no content class being assigned to the content under consideration, the primary content class equals a default class (see below).

Content Classes. Given the overall goals presented in Section 1.2, the following content classes are proposed. First, the content class *CC_AVAILABILITY* is assigned by applications that require content to be accessible given system dynamics. Second, the content class *CC_QUERY_EFFICIENCY* is used if content is required to be accessible with low latency. Third, the content class *CC_PERSISTENCE* is assigned if content must not be lost given transient and permanent failures of smart products. Finally, the content class *CC_DEFAULT* is assumed to be used internally by the distributed storage service if no content class is specified during storage and retrieval operations of initiators and requesters, respectively.

Note that the proposed content classes express *qualitative properties* only. Applications can only specify a single primary goal per operation and it is up to the applied content placement strategy to adapt number and placement of content-related replicas in a way that achieves the set of content associated goals to the best extent possible. Applications are not able to define quantitative properties such as maximum access latency or specific availability levels known from Service Level Agreements (SLA). Moreover, while the proposed classes show interdependencies, they all address a specific primary objective, only, according to the description of content properties presented in Section 2.1.3. Persistence does not imply availability, which in turn does not imply efficient access. Hence, defining distinct classes for the three content properties is in fact reasonable.

The novel replication strategy CC as well as the novel replacement strategy ECR presented in Sections 4.2.2 and 4.3.2, respectively, both utilise content classes in order to enhance replica organisation. They assume a distributed storage service that supports the above-stated functionality and operate with the content classes summarised in Table 4.8.

Content Class	Purpose
CC_AVAILABILITY	content is accessible when needed, but does not need to be accessible with low latency
CC_QUERY_EFFICIENCY	content is assessible with low latency, but does not need to be availabe all the time
CC_PERSISTENCE	content is not lost, but does not need to be available all the time
CC_DEFAULT	no optimisation

Table 4.8: Overview of the Proposed Content Classes

4.2.2 Replication Strategy: Content Class (CC)

The replication strategy CC makes use of the content classes described before to make informed replication decisions. For this purpose, the replication strategy assumes smart products to maintain both neighbours and content repositories that can be used as potential destinations for placing replicas (see Section 2.3.1). CC handles both initiator- and requester-assigned content classes and provides case-specific functionality depending on the role played by the smart product triggering and the one serving the operation as well as the operation-assigned content class.

CC_AVAILABILITY. In case an initiator classifies its storage operation with content class *CC_AVAILABILITY*, the primary provider tries to distribute replicas to a number of Ω distinct content repositories. The parameter Ω can be configured according to the properties of smart product application scenarios and enables balancing of replication degree and overall supplied availability. According to the definition of content repository nodes, i.e., nodes enabling reliable content location and retrieval (see Section 2.3.1), this approach can be assumed to increase content availability. Furthermore, especially in P2P storage systems, placement of replicas in the content repository set of primary providers is a common procedure for increasing content availability in the presence of node churn. The replication strategies Past Replication and OceanStore Put presented in Sections 3.1.3.3 and 3.1.3.1, respectively, can be taken as examples.

Primary providers apply a *replica maintenance mechanism* to address system dynamics that may affect content availability. When receiving content requests classified as *CC_AVAILABILITY*, primary providers exchange messages with the first Ω content repositories of their content repository set, which report whether they provide a replica of the given content. If the number of content repositories providing replicas of the given content falls below a number of $\lceil \frac{\Omega}{2} \rceil$, then the primary provider tries to place replicas on another set of $\lceil \frac{\Omega}{4} \rceil$ content repositories. This request-driven approach is favoured over a periodic maintenance, because it takes into account content popularity. Hence, the

actual level of availability of a content object depends on the number of requests the primary provider of the latter receives.

Note that the reduced creation of new replicas ($\lceil \frac{\Omega}{4} \rceil$ instead of the missing $\lceil \frac{\Omega}{2} \rceil$) is reasoned in terms of system dynamics. For example, the primary provider might not receive the aforementioned reporting from all of the addressed content repositories because of message timeout properties or temporary unavailability of content repositories. Consequently, while the reduced creation of new replicas during replica maintenance might result in reduced content availability in some cases, it prevents increasing replication degrees, which is important in resource-constrained smart product systems.

When a requester receives content of a request classified as *CC_AVAILABILITY*, it tries to create a local transient replica if the content-related access rate exceeds a pre-defined threshold Γ. Based on the assumption of temporal locality, this enhances availability independent from connectivity and state of the primary provider and its content repositories.

CC_QUERY_EFFICIENCY. Storage requests of initiators classified as *CC_QUERY_EFFICIENCY* are handled as storage requests of class *CC_AVAILABILITY* but with the configurable parameter $\Psi < \Omega$. In addition, if the initiator does not also play the role of primary provider for the given content, it attempts to (i) create a local transient replica and (ii) place a number of Φ transient replicas on the best neighbours of its neighbour set. While the number of replicas placed on-board of the content repositories of the primary provider preserves a minimum level of availability, the transient replica stored on-board of the initiator targets low-latency access. Moreover, the transient replicas distributed in the neighbour set of the initiator further address this objective given the potential storage limitation of smart products playing the role of initiator as well as the applied replacement strategy.[34] According to the life cycle of transient replicas depicted in Figure 4.11, there is no need for functionality that maintains number and location of transient replicas (remember that transient replicas are only maintained if they are accessed within their TTL).

Primary providers receiving content requests of class *CC_QUERY_EFFICIENCY* adopt the procedure applied for requests of class *CC_AVAILABILITY* but with the parameter $\Psi < \Omega$. If the requester receives the requested content and the number of requests exceeds a pre-defined threshold, it not only tries to place a transient replica on-board (see handling of content class *CC_AVAILABILITY*), but moreover on the best Φ nodes of its neighbour set.

[34] Similar to the workflow-based replication strategy CPA, this approach requires a distributed storage service that routes requests classified as *CC_QUERY_EFFICIENCY* to products in the environment of the requester, i.e., to the neighbours of the requester. Else, the placement of transient replicas in the neighbour set of a requester would not add any value. A potential implementation of this request routing is described in Section 5.3.4.4.

Content Class	Assigned by Initiator	Assigned by Requester
CC_AVAILABILITY	- Placement of replicas on a number of Ω distinct content repositories of the primary provider	- Request-driven replica maintenance by primary provider - Placement of a transient replica on-board of the requester if request rate exceeds threshold Γ
CC_QUERY_EFFICIENCY	- Placement of replicas on a number of Ψ < Ω distinct content repositories of the primary provider - Placement of a transient replica on-board of the initiator (if initiator ≠ primary provider) - Placement of transient replicas on a number of Φ distinct neighbours of the initiator (if initiator ≠ primary provider)	- Request-driven replica maintenance by primary provider - Placement of a transient replica on-board of the requester if request rate exceeds threshold Γ - Placement of transient replicas on a number of Φ distinct neighbours of the requester if request rate exceeds threshold Γ
CC_PERSISTENCE	- Erasure code replication	--
CC_DEFAULT	--	--

Table 4.9: Overview of the Replication Strategy Content Classes

CC_PERSISTENCE If storage requests of class *CC_PERSISTENCE* are served by a primary provider, then the latter applies erasure code replication. Given a sufficient level of erasure code redundancy, this approach is known to serve high levels of persistence (see replication strategy OceanStore Deep Archival Storage described in Section 3.1.3.1).

CC_DEFAULT Content requests of class *CC_PERSISTENCE* as well as storage and retrieval operations classified as *CC_DEFAULT* do not result in any replication operation. While persistence is covered solely when content is stored initially, content class *CC_DEFAULT* is used only in case applications are not aware of the purpose of the content they are requesting. An overview summarising the functionality of the replication strategy CC is presented in Table 4.9.

4.3 Replacement Strategies for Smart Products

This section presents the two novel replacement strategies MFRTR and ECR that utilise the concept of transient replicas with lazy removal properties and content classes, respectively. Note that both strategies cover the determination of *replacement candidates*, i.e., content to be replaced in order to serve a pending storage request, only. They do not represent self-contained replacement frameworks that deal with the actual process of moving replacement candidates to new providers. A replacement framework that enables plugging-in different strategies for determining replacement candidates is presented in Section 5.3.4.6.

4.3.1 Replacement Strategy: MFR for Transient Replicas (MFRTR)

The strategy MFR proposed by [KRT06, KRT07] maintains access rates for all content that was (directly or indirectly) requested within a configurable period of time. The strategy tries to keep content with high ratios between access rate and content size stored on-board, while using content with low ratios as potential replacement candidates (see Section 3.1.3.7). [KRT06, KRT07] claim MFR to achieve nearly optimal number and placement of replicas. The proposed replacement strategy MFRTR extends MFR by taking into account the life cycle of transient replicas depicted in Figure 4.11.

Concepts reused from MFR. According to MFR, MFRTR sorts content stored on-board in ascending order of the ratio between access rate and content size. Content objects with identical ratios are sorted in ascending order of the point in time of their last access. Consequently, content at the tail of the sorted queue is assessed most valuable driven by high access rates and / or small content size taking into account the limited storage capacity of smart products. Following MFR, MFRTR tries to keep content at the tail of the sorted queue stored on-board, while objects at the head of the sorted queue are taken as potential replacement candidates.

Novel Concepts of MFRTR. While MFR processes the sorted queue directly for determining replacement candidates, MFRTR leverages *lazy removal properties of transient replicas*. Prior to determining the actual set of replacement candidates D_r given the storage capacity required to serve a pending storage request (i.e., the difference of the size of the content to be stored and the available storage capacity of the to-be provider), MFRTR calculates the storage capacity s_a that could be made available by removing transient replicas flagged to be removed.

If the calculated storage capacity s_a is greater than or equal to the storage capacity required to serve the pending storage request, then the minimum number of transient replicas flagged to be removed needed to serve the request is deleted according to the above ordering. Consequently, in this case, the pending storage request is served without any replacement of content stored on-board.

Else, if the calculated storage capacity s_a is less than the storage capacity required to serve the pending storage request, then all transient replicas flagged to be removed are taken off the sorted queue. This shortened queue is processed head to tail in order to determine the set of replacement candidates D_r. If these replacement candidates are replaced successfully by the replacement framework, then the transient replicas identified before are removed eventually. Hence, this case covers both deletion of transient replicas flagged to be removed and content replacement in order to serve a pending storage request.

Consequently, by making use of the lazy removal properties of transient replicas, MFRTR reduces the number of content to be replaced for serving pending storage requests. Compared to MFR, this leads to lower replacement and communication overhead unless there are no transient replicas flagged to be removed available on the node operating the strategy. In this case, MFRTR falls back to MFR. The strategy is summarised in the form of pseudo code in Algorithm 4.1.

Algorithm 4.1: Replacement Strategy: MFR for Transient Replicas (MFRTR)

```
1:  procedure MFRTR(q, d)          ▷ free storage capacity q for storing new content d
2:      sort content stored on-board using MFR
3:      a ← calculate storage capacity that could be made available by removing tran-
    sient replicas flagged to be removed
4:      if q ≤ a then
5:          remove required transient replicas flagged to be removed      ▷ MFR ordering
6:          return
7:      else
8:          q = q − a
9:          determine replacement candidates D_r                          ▷ MFR ordering
10:         relocate replacement candidates D_r
11:         if replacement candidates are replaced successfully then
12:             remove transient replicas flagged to be removed
13:             return
14:         else
15:             return                                  ▷ d cannot be stored on-board
16:         end if
17:     end if
18: end procedure
```

4.3.2 Replacement Strategy: Enhanced Content Replacement (ECR)

MFRTR relies on MFR to determine the set of replacement candidates D_r if a pending storage request cannot be served by deleting transient replicas flagged to be removed, only (see Algorithm 4.1, line 9). This functionality is adapted by the replacement strategy ECR. ECR provides a content-class-aware concept for determining replacement candidates taking into account the content class associated with the pending storage request as well as the primary content classes of the content stored on-board. This way, ECR complements the proposed replication strategy CC and fosters maintenance of content properties as described in Section 4.2.1.

Location Quality Ordering. ECR makes use of the location properties of the defined content classes. It is based on the transitive location quality ordering *CC_DEFAULT* $<$ *CC_PERSISTENCE* $<$ *CC_AVAILABILITY* $<$ *CC_QUERY_EFFICIENCY* from the perspective of the smart product executing the replacement strategy. Thus, content of class *CC_QUERY_EFFICIENCY* is tried to be bound to the product as long as possible to ensure low-latency access. Content of class *CC_AVAILABILITY* is less location-dependent and is rather replaced than content of class *CC_QUERY_EFFICIENCY*. Even further, content of class *CC_PERSISTENCE* is not location-dependent and is favoured over content of class *CC_AVAILABILITY* when determining replacement candidates. As described in Section 4.2.2, content of class *CC_PERSISTENCE* is replicated by means of erasure code replication. Hence, the level of persistence is affected by the degree of erasure code redundancy rather than the location of the resulting erasure-coded blocks. Finally, content of class *CC_DEFAULT* is best suited for replacement because of its indifferent classification (remember that this class is used internally by the distributed storage service, only, and cannot be assigned by applications).

Replacement of Content with Lower Location Quality. Based on this location quality ordering, ECR applies a case-by-case strategy for determining replacement candidates. In order to *preserve location properties* to the best extent possible, ECR tries to replace content with a location quality lower than the content class associated with the pending storage request. Thus, ECR searches for content of class *CC_DEFAULT* in a first step and incrementally increases location quality, if needed. For example, if a storage request is assigned content class *CC_AVAILABILITY*, ECR initially tries to make available the storage capacity required to serve the pending storage request by candidates of primary content class *CC_DEFAULT*. If this does not free the required storage capacity, then ECR searches for content of primary content class *CC_PERSISTENCE* in a next step and extends the set of replacement candidates, accordingly. Note that the basic ordering in which replacement candidates are searched for still complies with the MFR ordering MFRTR is based upon.

Replacement of Content with Equal Location Quality. If not enough storage capacity can be made available by replacement candidates with lower location quality, then ECR searches for content with primary content class equal to the class associated with the pending storage request. In this case, ECR uses the metric $f^{ECR}(d, D_r, T)$ presented in Eq. 9 to compare the benefit of storing the new content d on-board with the cost of replacing the subset D_r of the replacement candidates that includes candidates with primary content class matching the classification of d, only. Again, similar to the metrics used by the proposed workflow-based replication strategies, the parameter T refers to the current period and is used to enable modelling of temporal locality properties.

Benefit Calculation. The metric $b^{ECR}(d, T)$ for assessing the value of storing the new content d given the current period T is presented in Eq. 10. It incorporates the request rate r_t^d of content d in period t, which is weighted by the period in which requests took place. The weighting factor equals the approach used by the proposed workflow-

$$f^{ECR}(d, D_r, T) = \frac{b^{ECR}(d, T)}{c^{ECR}(D_r, T)}) \tag{9}$$

$$b^{ECR}(d, T) = \begin{cases} \Theta & \text{if } \forall t \in \{1..T\} : r_t^d = 0 \\ \sum_{t=1}^{T} \frac{r_t^d}{T-t+1} \times \frac{1}{s_d} & \text{else} \end{cases} \tag{10}$$

$$c^{ECR}(D_r, T) = \sum_{d' \in D_r} \sum_{t=1}^{T} \frac{r_t^{d'}}{T-t+1} \times \frac{1}{s_{d'}} \times \frac{1}{e_{d'}} \tag{11}$$

d : new content to be stored
D_r : set of replacement candidates
T : current period
$r_t^d, r_t^{d'}$: number of requests for content d, d' in period t
$s_d, s_{d'}$: size of content d, d'
$e_{d'}$: time elaspsed since last access of content d'

based replication strategies (see Section 4.1.2.1). The resulting weighted request rate is divided by the size s_d of the content d. Consequently, the benefit of storing the new content increases with the number of requests occurred in past periods and decreases with the size of the new content.

Note that the proposed replication strategies PA and CPA incorporate content size in the cost metric, only (see Section 4.1.3.1). Yet, unlike PA and CPA, the benefit and cost calculation of ECR cover different content (the new content and potential replacement candidates). Moreover, the content size used in the benefit metric $b^{ECR}(d, T)$ may be significantly greater than the sum of the content sizes of the replacement candidates D_r covered in the cost metric explained below. This is due to the set D_r consisting of content of the same class as the new content d, only; replacement candidates of lower location quality are not covered by the metric $f^{ECR}(d, D_r, T)$.

If the new content was not accessed in any period, i.e., $\forall t \in \{1..T\} : r_t^d = 0$, the benefit metric is set to a configurable default value Θ. This is in particular needed for content being pre-replicated by the proposed workflow-based replication strategies.

Cost Calculation. Similar to the benefit metric, the cost metric $c^{ECR}(D_r, T)$ calculates the weighted request rate divided by the content size for each replacement candidate $d' \in D_r$ given the current period T (see Eq. 11). Moreover, the weighted requested rate is divided by the time $e_{d'}$ elapsed since the last access of content d' to further strengthen impact of temporal locality properties of access patterns. The resulting value is summed

up over all replacement candidates $d' \in D_r$ in order to determine the overall replacement cost. Note that for limiting the affect of the parameter $e_{d'}$, it might make sense to use pre-defined intervals instead of the actual time delta. An exemplary configuration of $e_{d'}$ is presented in Section 5.4.1.2.

Result. The new content d is only stored on-board if the benefit $b^{ECR}(d, T)$ of storing d outweighs the cost $c^{ECR}(D_r, T)$ of replacing the replacement candidates $d' \in D_r$, i.e., $f^{ECR}(d, D_r, T) \geq 1$, and – certainly – if enough storage capacity can be made available. The replacement strategy ECR is summarised in the form of pseudo code in the Algorithms 4.2 and 4.3. Note that for reasons of readability, the pseudo code is limited to the content classes *CC_DEFAULT*, *CC_AVAILABILITY*, and *CC_QUERY_EFFICIENCY*. Yet, the pseudo code can be easily extended by content class *CC_PERSISTENCE*, which is processed in a way equivalent to the other classes.

Algorithm 4.2: Replacement Strategy: Enhanced Content Replacement (ECR) (1/2)

```
1:  procedure DETERMINEREPLACEMENTCANDIDATES(q, d)        ▷ determine replacement
    candidates to free storage capacity q for storing new content d
2:      switch content class of d do
3:          case DEFAULT
4:              a ← calculate storage capacity that could be made available by replacing
    content D_r^DEF of class DEFAULT
5:              if q ≤ a AND f^ECR(d, D_r^DEF, T) ≥ 1 then
6:                  return replacement candidates D_r^DEF
7:              else
8:                  return                                ▷ d cannot be stored on-board
9:              end if
10:         case AVAILABILITY
11:             a ← calculate storage capacity that could be made available by replacing
    content D_r^DEF of class DEFAULT
12:             if q ≤ a then
13:                 return replacement candidates D_r^DEF
14:             else
15:                 q = q − a
16:                 a ← calculate storage capacity that could be made available by replac-
    ing content D_r^AVA of class AVAILABILITY
17:                 if q ≤ a AND f^ECR(d, D_r^AVA, T) ≥ 1 then
18:                     return replacement candidates D_r^DEF ∪ D_r^AVA
19:                 else
20:                     return                            ▷ d cannot be stored on-board
21:                 end if
22:             end if
```

Algorithm 4.3: Replacement Strategy: Enhanced Content Replacement (ECR) (2/2)

```
23:       case QUERYEFFICIENCY
24:           a ← calculate storage capacity that could be made available by replacing
      content D_r^DEF of class DEFAULT
25:           if q ≤ a then
26:               return replacement candidates D_r^DEF
27:           else
28:               q = q − a
29:               a ← calculate storage capacity that could be made available by replac-
      ing content D_r^AVA of class AVAILABILITY
30:               if q ≤ a then
31:                   return replacement candidates D_r^DEF ∪ D_r^AVA
32:               else
33:                   q = q − a
34:                   a ← calculate storage capacity that could be made available by
      replacing content D_r^QUE of class QUERYEFFICIENCY
35:                   if q ≤ a AND f^ECR(d, D_r^QUE, T) ≥ 1 then
36:                       return replacement candidates D_r^DEF ∪ D_r^AVA ∪ D_r^QUE
37:                   else
38:                       return                          ▷ d cannot be stored on-board
39:                   end if
40:               end if
41:           end if
42: end procedure
```

4.4 Deployment Options

As described in Section 2.2.1, smart products may considerably vary in terms of their on-board storage resources as well as their computation and communication capabilities. Given a specific scenario, this might affect applicability of the proposed content placement strategies.

Complexity of CC and MFRTR. While both the replication strategy CC and the replacement strategy MFRTR require rather basic computation complexity, they may lead to a high number of replicas and – consequently – an increased usage of the overall storage capacity of smart product systems. Yet, the number of replicas and transient replicas distributed and maintained by CC can be controlled by the configuration parameters Ω and Ψ explained in Section 4.2.2, which have to be set according to the resources of the smart products participating in the target scenario. The evaluation study presented in Section 5.4 covers multiple parameter settings and exemplifies the impact of different

replication degrees effected by CC on the achieved level of availability and query efficiency. Also, while the lazy removal properties of transient replicas yield a higher average replica population, this does not cause storage issues or increased network load caused by replacement operations given an application of either MFRTR or ECR. In fact, lazy removal of transient replicas fully exploits available storage capacity of smart products to enhance content availability and query efficiency.

Complexity of MPP, PA, CPA, and ECR. In contrast, the workflow-based replication strategies MPP, PA, and CPA as well as the content-class-aware replacement strategy ECR include calculations with nested summations and require more computation capabilities than CC and MFRTR. For example, this applies to ECR in case content with primary content class equal to the class associated with the pending storage request is to be replaced. The corresponding algorithm of the metric $f^{ECR}(d, D_r, T)$ presented in Eq. 9, which is referred to in lines 5, 17 and line 35 of the Algorithms 4.2 and 4.3, respectively, is of polynomial time with an exponent of 2 (i.e., $O(n^2)$). Similarly, while all three workflow-based replication strategies may require multiple iterations depending on the workflow structure, the path assessment metrics of PA and CPA (see $f^{PA}(p, T)$ and $f^{CPA}(p, T)$ presented in Eq. 2 and Eq. 6, respectively) include nested summations and are both algorithms of polynomial time. Moreover, driven by potential pre-replication wastage, the three workflow-based replication strategies consume additional storage capacity. Finally, CPA generates extra network traffic resulting from the regular index exchange between smart products and their surroundings. Given the increased computation complexity, storage use, and network load, it may be required to analyse different deployment options of the above strategies.

Heterogeneity in Smart Product Systems. In the scenarios analysed in the course of the EU-funded research project SmartProducts, workflow-based operations are typically controlled by *primary smart products*. Regardless of the proposed workflow-based replication strategies, primary smart products execute workflows by means of a workflow engine in order to process activities and retrieve activity-related content needs (see Figure 4.9(a)). In case of distributed workflows, the primary smart product moreover takes care of the orchestration of sub workflows processed by surrounding nodes.

For example, in the smart aircraft manufacturing scenario described in Section 2.2.3, workflows are processed only by nomadic devices, which can be assumed to possess enough resources to execute the proposed strategies. Other smart products involved in workflow execution such as smart wrenches or smart drills merely process single tasks. While they make use of a locally deployed – potentially lightweight version of a – distributed storage service to retrieve and store information such as configuration settings or work results, they do not carry on an instance of the proposed workflow-based replication strategies.

Deployment Options. In case the primary smart product processing the workflow is not capable of operating the proposed workflow-based replication strategies (e.g., because it runs out of energy or is occupied with other tasks), alternative deployment

options are to be taken into consideration. The workflow-based replication strategies could be operated in the local infrastructure such as the workstations participating in the smart aircraft manufacturing scenario. The primary smart product would merely operate a proxy / client in order to invoke processing of the replication strategy operated by the local infrastructure. Similarly, one could imagine smart products being organised in clustered, hierarchical overlay network structures as proposed by [MEA+11]. In this case, local cluster heads could take over operation of the proposed replication strategies. Finally, depending on the communication capabilities of the smart product targeting workflow execution, it could also be conceivable to fully outsource operation of the proposed replication strategies into the cloud (see Section 2.3).

Each of these three cases reduces computation complexity for primary smart products targeting workflow processing. Even further, the deployment options could increase efficiency of the proposed replication strategies in scenarios with access patterns following geographic locality. Thus, the entity operating the replication strategies (i.e., the local infrastructure, a cluster head, cloud infrastructure) could aggregate knowledge of connected smart products. This information could be leveraged to enable more accurate cost / benefit calculation taking into account content interests of connected smart products. This enhancement could compensate the potential loss in query efficiency in case of pre-replicated content being stored on-board of the aforementioned entities instead of primary smart products.

Finally, in the worst case of a smart product targeting workflow processing not being capable of operating the proposed replication strategies and no option being available for outsourcing the latter, workflow execution would merely fall back to traditional on-demand content retrieval as presented in Figure 4.9(a).

4.5 Requirements Mapping

Classification. Given the classification scheme defined in Section 3.1.2, both MPP and PA are categorised as requester-initiated, active strategies that use local knowledge to place an entity-specific number of replicas on the requester side during workflow execution (i.e., on-demand). CPA adapts PA by incorporating content interest of surrounding smart products. Hence, CPA employs a cooperative sphere of knowledge. Finally, CC is a requester- and initiator-triggered reactive strategy that uses local knowledge to create a content-class-specific number of replicas (uniform per content class). Depending on the content class, replicas are either placed on the provider side, or on the provider as well as the requester side. This one-time replication is complemented by a request-driven replica maintenance concept (on-demand). The resulting classification is presented in Table 4.10.

Replication Strategy	Replica Granularity	Replication Degree	Replication Schedule	Replication Initiation	Replica Placement	Sphere of Knowledge	Activity History
MPP	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Active
PA	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Local	Active
CPA	Partial	Entity-Specific	On-Demand	Requester	Requester-Side	Cooperative	Active
CC	Partial	Uniform	One-Time, On-Demand	Initiator, Requester	Requester-Side, Provider- Side	Local	Reactive

Table 4.10: Classification of the Proposed Replication Strategies

Replication Strategy	R1	R2	R3	R4	R5	R6	R7	R8
MPP	●	○	●	●	●	○	●	●
PA	●	○	●	●	●	○	●	●
CPA	●	●	●	●	●	○	●	●
CC	●	○	○	◍	●	●	●	●

Table 4.11: Mapping of Requirements and the Proposed Replication Strategies

Requirements Mapping. With regard to the requirements defined in Section 2.4, all four proposed replication strategies address both the *resource limitation* (R7) and the *scalability challenges* (R8) of smart product systems. A separate analysis of the message and computation complexity of the approaches as well as different deployment variants for supporting scenarios with resource-constrained smart products are presented in Section 4.4. Also, all strategies provide considerable *configuration options* (R5). An analysis of different parameter settings is described as part of the evaluation study in Section 5.4.1. Hence, the requirements R5, R7, and R8 are clearly supported by the proposed replication strategies.

Workflow-based Replication. The three workflow-based replication strategies incorporate temporal locality properties for determining path execution probability as well as for calculating benefit and cost of content pre-replication. Moreover, they base upon workflow models that represent a subset of problem-solving knowledge associated with smart products (see Section 2.2.2) and target improvement of guided product-to-user interaction by means of optimised content access. For this reason, all three workflow-based replication strategies address the requirements R1, R3, and R4. Also, compared to related work assessed in Section 3.2, the three proposed strategies cope with control flow uncertainty and provide means for balancing improvement of query efficiency with pre-replication wastage. Even further, the cooperative strategy CPA utilises content interest of surrounding smart products in order to leverage the geographic locality properties of access patterns in the domain of smart products described in Section 2.2.2.2. Hence, CPA supports requirement R2 as well.

Content-Class-based Replication. In contrast, CC does not support any of the requirements R2, and R3. Yet, the threshold-based replica placement on the requester side

makes use of temporal locality properties (R1). Most important, with the introduction of content classes, CC provides applications with simple means for classifying content and adapting content organisation, accordingly. Moreover, as opposed to related work presented in Section 3.2, CC is not limited to a single objective but supports all content properties defined in Section 2.1.3. Finally, in case of content class *CC_QUERY_EFFICIENCY*, CC also targets enhancement of query efficiency and can be rated as partly supporting requirement R4.

Conclusion. In conclusion, as to the knowledge of the author, the four proposed replication strategies are the first that address the defined requirements while taking into account the challenges of the domain of smart products. While none of the proposed strategies addresses all requirements on its own, it is the combination of CPA and CC that fully meets the defined requirements. This is further supported by the two replacement strategies MFRTR and ECR, which cover the lazy removal properties of transient replicas and the concept of content classes, respectively, and complement the proposed replication strategies. This way, the novel integrated content placement strategies address the entire replication life cycle (see Section 2.1.2) in the domain of smart products. An overview of the mapping of requirements and the proposed strategies is presented in Table 4.11.

4.6 Chapter Summary

This chapter presented the contribution of this thesis. This includes the three workflow-based replication strategies MPP, PA, and CPA, as well as the related concept of transient replicas. By leveraging the procedural knowledge associated with smart products in the form of workflows, the three strategies predict and pre-replicate upcoming activity-related content needs. While all pursue the same objective and apply concepts commonly applied by replication strategies, they differ (i) in the way of balancing improvement of content organisation with the risk of false pre-replication as well as (ii) the incorporated sphere of knowledge. The concept of transient replicas addresses the active procedure of the proposed strategies and limits the overall number of replicas to be maintained.

Moreover, the concept of content classes is presented and applied by the replication strategy CC. This way, CC provides smart product applications with an explicit means for specifying content use, which is utilised for improving replica organisation. The four replication strategies are complemented by the two replacement strategies MFRTR and ECR, which cover the life cycle of transient replicas and the concept of content classes.

Given the limited resources of smart products, this chapter analysed the complexity of the proposed strategies and discussed different deployment options to point out the

applicability of the strategies in the domain of smart products. Also, the novel replication strategies were classified based on the schema presented in Section 3.1.2 and a mapping to the requirements presented in Section 2.4 was conducted. This analysis revealed that the proposed strategies are the first that address the defined requirements while taking into account the challenges of smart products. They cover the entire replication life cycle and enable optimisation of content availability, persistence, and query efficiency. The actual improvement of the content placement strategies is analysed in the following chapter by means of a simulation-based evaluation study.

In future work, the proposed content placement strategies could be enhanced in the following directions. First, the general assumption of immutable content could be addressed by a consistency maintenance strategy. Also, the effect of integrating content update rates in the metrics used by the proposed strategies could be analysed. Second, the inherent heterogeneity of smart products could be taken into account for enabling dynamic parameter configuration based on available storage capacity and upcoming content needs. Third, the strategy CC could be extended to enable the definition of quantitative content properties (e.g., maximum access latency) instead of the current approach of using qualitative categorisation and best effort strategies. Finally, while the proposed strategies are not tailored to any (overlay) network architecture to ensure their generic applicability, adaptations towards specific network architectures could be analysed in future work. A detailed description of these topics is presented in Section 6.2.

Chapter 5
Implementation and Evaluation

While functionality of smart products such as context awareness and processing or integration with back-end systems can typically be analysed and assessed by means of small-scale experiments, the evaluation of complex distributed services requires extensive test environments. Especially in early research and development phases, the installation of large test beds is not appropriate. Therefore, simulation environments appear to be the approach of choice.

For this reason, a simulation-based evaluation study is contributed in this chapter to assess the proposed content placement strategies. This assessment includes the extent to which the different strategies address the overall objectives stated in Section 1.2 as well as the trade-off between this enhancement and their complexity and cost. Moreover, the simulation-based evaluation study covers comparison of the proposed content placement strategies with representatives of related work.

The chapter is structured as follows. Section 5.1 introduces the methodology underlying the simulation-based evaluation study with its four phases system analysis, simulation modelling, simulation execution, and result analysis. Thereafter, Section 5.2 presents the result of the *system analysis phase*, the conceptual simulation model. This includes the description of the system layers captured by the simulation model (see Section 5.2.1), simulation events (see Section 5.2.2) and parameters (see Section 5.2.3), as well as quality attributes used to evaluate the proposed strategies (see Section 5.2.4). Moreover, Section 5.2.5 presents the modelling of the industry scenario smart aircraft manufacturing described in Section 2.2.3, which is used as reference scenario for the evaluation study. Next, Section 5.3 describes the outcome of the *simulation modelling phase*, the implemented simulation model. This includes high-level design descriptions covering both the compositional and the dynamic structure of the implemented simulation model, as well as detailed design descriptions of the main components. The implementation of related work is presented in Section 5.3.6. Finally, the results of the phases *simulation execution* and *result analysis* of the applied methodology are described in Section 5.4. The section covers the configuration of the simulation framework and the proposed strategies (see Section 5.4.1), as well as the analysis of simulation data given the quality attributes defined in the conceptual simulation model (see Sections 5.4.2 and

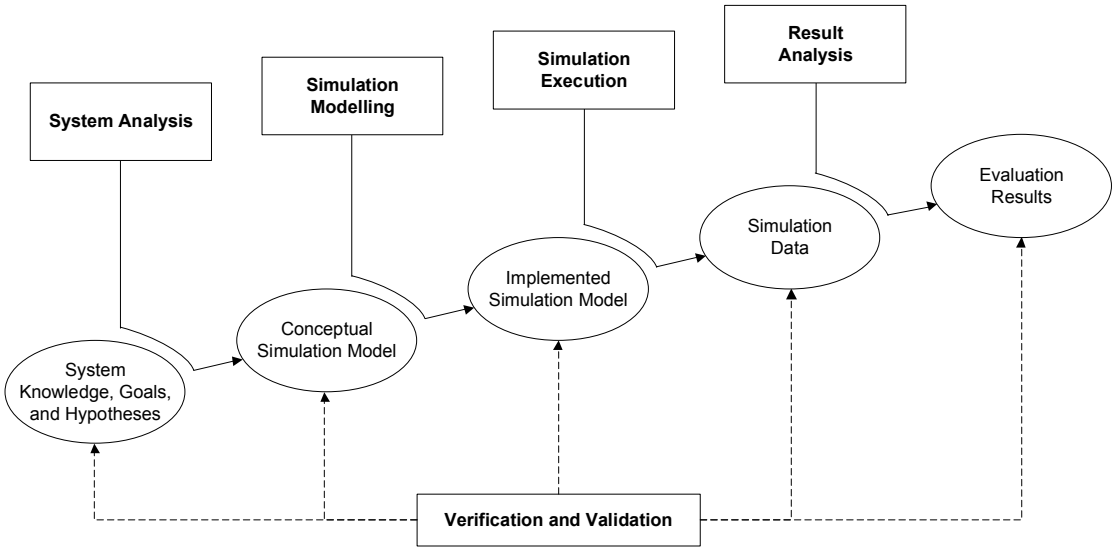


Figure 5.13: Modelling and Simulation Process (based on [Kö01])

5.4.3). The chapter concludes in Section 5.5 with a summary of the main results of the evaluation study.

This chapter is based upon and partly reuses descriptions out of the following publications of the author of this thesis: [MBGB11]. Note that paragraphs reused from these publications are not marked as direct quotes if they were written exclusively by the author of this thesis. Quotations within reused paragraphs are taken over and marked accordingly.

5.1 Methodology

The methodology applied to conduct the simulation-based evaluation adopts the iterative modelling and simulation process proposed by [Kö01]. As depicted in Figure 5.13, this process consists of four phases, which are described in the following paragraphs and reflected in the structure of this chapter.

System Analysis. In phase 1, *System Analysis*, the system to be simulated is analysed with respect to the defined system goals. This encompasses the definition of system boundaries and the determination of relevant system components including their relationship and dependencies. Even further, simulation parameters, quality attributes, as

well as evaluation scenarios and data are defined. Phase 1 results in the conceptual simulation model.

Simulation Modelling. In phase 2, *Simulation Modelling*, the conceptual simulation model is formalised and transformed into the implemented simulation model, also referred to as computer model. This phase includes the selection and potential extension of a suitable simulation framework as well as the implementation of both the strategies to be evaluated and the components they depend upon as defined in the conceptual simulation model.

Simulation Execution. In phase 3, *Simulation Execution*, the implemented simulation model is instantiated and executed with different data sets and / or configurations (e.g., network size, random seed values). This enables detailed evaluation including determination of average values for the quality attributes defined in the conceptual simulation model.

Result Analysis. Finally, the results of the simulation runs are analysed with respect to the overall simulation goals using the quality attributes defined afore (e.g., to which extent do the proposed content placement strategies enhance query efficiency and content availability in smart product systems) in phase 4, *Result Analysis*.

The continuous verification of simulation model transformations as well as the validation of the simulation model with respect to the (assumptions of the) real-world system and the system goals stresses the non-linearity of the modelling and simulation process.

5.2 Conceptual Simulation Model

In the ubiquitous computing domain as well as related research areas, simulation frameworks are typically classified into *network simulators* and *overlay simulators*. On the one hand, network simulators such as NS-3[35] or OMNeT++[36] facilitate packet-level simulation of network protocols and can be utilised for analysing wireless sensor networks or RFID systems. On the other hand, overlay simulators such as OverSim[37] or PlanetSim[38] abstract from network details and focus on overlay network routing protocols and services [ALA+08]. However, while both kinds of simulators could be used as basis for simulating and evaluating content placement strategies for smart products, there is no simulation framework and model that fully reflects the characteristics of smart product systems.

[35] http://www.nsnam.org/
[36] http://www.omnetpp.org/
[37] http://www.oversim.org/
[38] http://projects-deim.urv.cat/trac/planetsim/

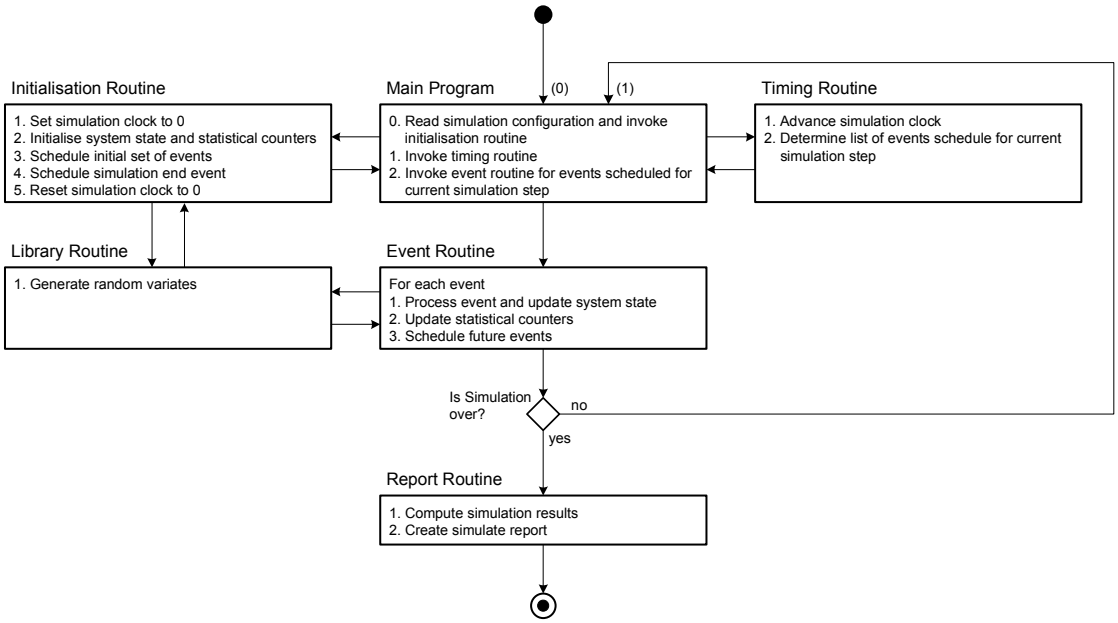


Figure 5.14: Simulation Control Flow of Discrete Event Simulations (based on [Law06])

Extended Overlay Simulation Model. In order to facilitate simulation of content placement strategies for smart products, an *extended overlay simulation model* is proposed. This model assumes smart products to be organised in P2P overlay networks and reflects many of their characteristics such as their heterogeneity, resource limitation, and workflow-based operations. As opposed to common overlay simulators that use simplified topologies, the extended overlay simulation model enables integration of underlay networks with realistic topologies in order to enhance simulation accuracy.

Discrete Event Simulation. The extended overlay simulation model is designed for *discrete event simulation* and follows the phases and the control flow defined by [Law06]. As depicted in Figure 5.14, discrete event simulation consist of three phases. First, in the *initialisation phase*, the simulation configuration is read and the initialisation routine is invoked to set up the initial system state, i.e., an initial amount of nodes randomly placed in the network as well as content being randomly assigned to these nodes.[39] Moreover, an initial set of events such as nodes joining or leaving the network as well as the simulation end event are scheduled. Second, in the *execution phase*, the scheduled events are processed. This includes (i) processing of the event logic (e.g., creation of a new node and adding of this node to the network by means of the join functionality of the overlay network implementation) and (ii) scheduling of associated events. This event processing approach is iterated upon until the simulation end event scheduled in

[39] Both placement and assignment follow the distribution rules of the applied evaluation scenario.

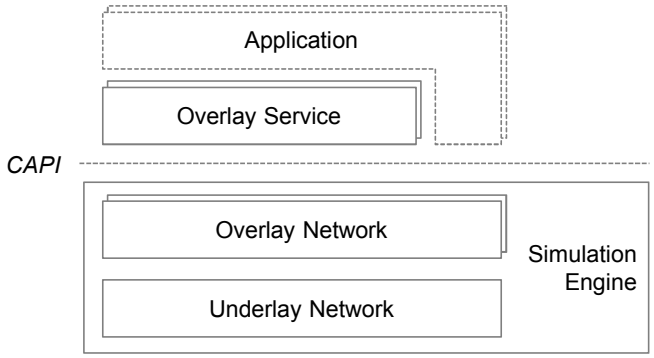


Figure 5.15: Conceptual Structure of the Simulation Model

the initialisation phase is reached. This event closes the execution phase and starts the third phase of a discrete event simulation, the *reporting phase*. This phase computes the simulation results based on the data maintained during simulation execution and creates simulation reports according to a pre-defined set of quality attributes.

5.2.1 Simulation Layers

The proposed extended overlay simulation model represents smart product systems in an idealised and abstracted way. Due to its clear purpose, the simulation model is focused on components and functionality relevant for evaluating the novel content placement strategies [BFS87]. The characteristics of smart product systems reflected in the simulation model include the heterogeneous storage capacity and communication capability of smart products, system dynamics, and content specifics such as content size and content availability requirements.

Underlay Network Layer. The layered structure of the simulation model is presented in Figure 5.15. The underlay network layer is used to model basic underlay networks with realistic topologies. This includes node distribution and clustering, bandwidth allocation, as well as communication delays. Due to the purpose of the simulation model, additional underlay network properties such as packet delay variation or packet loss rates are not considered. This way, quality attributes of content placement strategies such as average content access latency or hit rate given that messages are assigned predefined timeout properties can be evaluated with much higher accuracy compared to simulation models that purely rely on virtual random topologies.

Overlay Network Layer. The second layer, overlay network, encapsulates the overlay network built upon the underlay network topology as well as the functionality of the actual P2P content location and routing substrate.

Overlay Service & Application Layer. While most simulation models solely consider a single application layer, the proposed model distinguishes between overlay services, i.e., distributed services that directly operate on the P2P overlay network, and the actual application logic that utilises functionality of the overlay service or overlay network layer. According to the purpose of the simulation model, the overlay service layer covers both a distributed storage service encapsulating the proposed content placement strategies and a workflow engine that is needed to simulate the novel workflow-based replication concepts.

Common API. The simulation model applies the de-facto standard interface for structured overlay networks, the Common API (CAPI) as defined by [DZD+03]. This ensures clear separation of concerns and eases simulation of different overlay networks and services. While events scheduled and processed by the simulation engine affect all layers of the simulation model, only the two lower layers are actually controlled by the simulation engine and require simulator-specific implementations. In contrast, overlay services that comply with the Common API as well as the actual applications can directly be deployed in the simulation environment. This simplifies simulation of distributed services of smart products and is especially valuable if simulation is used in early development phases and complemented by test-bed experiments.

5.2.2 Simulation Events

Overlay Network Events. The dynamics of smart product systems are modelled by events that are scheduled and processed by the simulation engine. On the underlay and overlay network layer, the simulation model supports the typical events for simulating node churn: node *JOIN*, node *FAIL*, as well as node *LEAVE* events. For example, joining nodes are integrated into the underlay network, in which they are associated with unassigned underlay nodes that reflect their capabilities, and self-organise into the overlay network structure according to the means of the concrete overlay network implementation. Node *FAIL* events are used to model ungraceful leaves of nodes, i.e., nodes that suddenly disappear (e.g., because they run out of energy or loose connectivity). This behaviour is complemented by node *LEAVE* events, which are used to model graceful leaves of nodes that explicitly announce their leave to enable other nodes to react accordingly (e.g., by triggering content handover).[40]

Overlay Service Events. In addition, the simulation model covers overlay service events for simulating distributed storage services. This includes content *GET* and *PUT* events, which reflect requests for content retrieval and content storage, respectively.

[40] Move events capturing the potential mobility of smart products are not covered by the proposed simulation model. While simple approaches such as random node movement can be assumed to not provide additional insights, the incorporation of a mobility model considerably increases complexity of the simulation model.

Replication strategies are indirectly covered by these events and must be estimated separately in order to get an idea of the expected overall load generated during simulation execution. Since the proposed content placement strategies assume immutable content, the simulation model does not include events for simulating content update operations.

To enable simulation and evaluation of the proposed workflow-based replication strategies, the simulation model moreover includes *WORKFLOW* execution events. These events trigger workflow operations including related *GET* and *PUT* requests for collecting activity-related content needs and persisting workflow execution results, respectively.

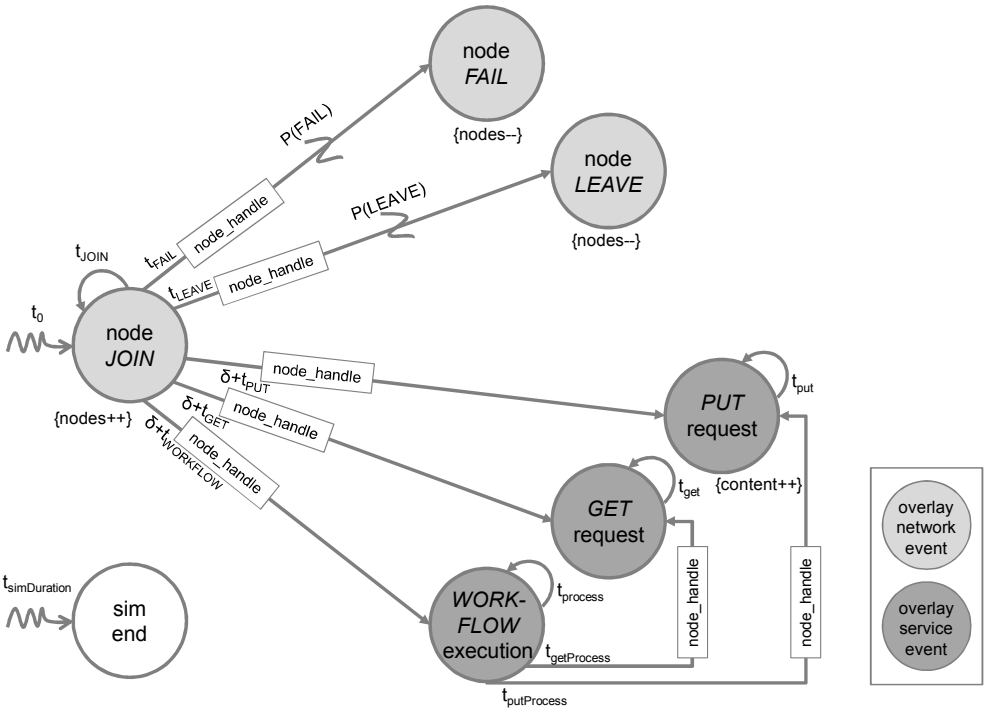


Figure 5.16: Node-Centric Simulation Event Graph

Node-Centric Event Scheduling. Both overlay network and overlay service events are scheduled according to a *node-centric event scheduling* approach. Hence, instead of having the simulation engine randomly assigning events to nodes, nodes are themselves responsible for scheduling their events. Figure 5.16 shows the applied simulation event graph based on the notation defined by [Bus96]. According to this graph, node *JOIN* events are scheduled during the simulation initialisation phase in order to prepare a basic network structure. *JOIN* events schedule node-specific *PUT*, *GET*, and *WORKFLOW* execution events as well as themselves to enable new nodes joining the network. Each of these events reschedules itself in order to simulate node-specific activities according to the distribution functions presented in Section 5.2.3.

Moreover, joining nodes have an initialisation delay of a fixed number of δ simulation steps in order to facilitate node stabilisation. Indeed, they can receive requests playing the role of intermediates / providers during the initialisation delay, but they don't play the role of requesters or initiators.

Finally, since *LEAVE* and *FAIL* events are exclusive, i.e., nodes may either fail or leave the network, the two events are assigned execution probabilities that are evaluated during the scheduling phase of the node *JOIN* event routine.

5.2.3 Simulation Parameters

5.2.3.1 Simulation Parameters of Simulation Events

The distribution functions of node *JOIN* events as well as content *GET* and *PUT* events comply with the distribution functions described in [KRT07, SW05]. Node *LEAVE* and *FAIL* events adopt the lifetime churn model defined by the simulation framework OverSim.[41] The exclusive scheduling of the two events is taken into account by the probability factor ρ. Finally, *WORKFLOW* execution events are scheduled based on a Gaussian distribution with absolute values being used to avoid negative scheduling times (see Table 5.12). Note that the actual configuration of these distribution functions may vary in order to reflect different node classes of smart product systems.

Simulation Parameter	Distribution Function	Comment
Node *JOIN*	t_{JOIN} := Poisson(λ)	Node interarrival time
Node *FAIL*	t_{FAIL} := Weibull(α,β) P(*FAIL*) = ρ	Based on LifetimeChurn model
Node *LEAVE*	t_{LEAVE} := Weibull(α,β) P(*LEAVE*) = 1 - P(FAIL) = 1- ρ	Based on LifetimeChurn model
GET request	t_{GET} := Poisson(λ)	
PUT request	t_{PUT} := Poisson(λ)	
WORKFLOW execution	$t_{WORKFLOW}$:= |Normal(μ,σ^2)|	Absolute value to avoid negative samples

Table 5.12: Simulation Parameter: Events

[41] http://www.oversim.org/wiki/OverSimChurn

5.2.3.2 Node and Content Parameters

In addition to simulation events, the proposed extended overlay simulation model includes node- and content-related simulation parameters. This enables modelling of heterogeneous node classes, which differ in the configuration of their on-board storage capacity. Heterogeneous communication capabilities are modelled by associating node classes with matching underlay nodes. For example, nodes with few resources are assigned to underlay nodes with low-bandwidth communication links. Moreover, multiple content types (not to be confused with content classes defined in Section 4.2.1) can be modelled and assigned to node classes, with each content type consisting of content of different sizes. This is required, because smart products with limited resources typically generate and request content of relatively smaller size compared to powerful smart products.

Node Storage Capacity and Content Size. In order to facilitate simulations with high numbers of nodes and content objects, both storage capacity and content size are defined in virtual storage units (i.e., Integer values). For both parameters, a finite-range discrete distribution function is applied to (i) randomly set storage capacity of nodes subject to the class they belong to as well as (ii) randomly select content of different size out of the content type associated with the chosen node class.

Content Access Popularity. Content popularity is configured using a Zipf-like distribution function [SW05]. In combination with the workflow-related content needs (see Sections 5.3.5 and 5.4.1.4), this configuration can be assumed to yield access patterns with temporal locality properties. Access patterns with geographic locality properties are not captured directly by the simulation model. They can be assumed to result from the content popularity configuration and the spatial clustering of smart products described below.

Activity Duration. Finally, for all workflows, a consistent minimum activity execution time is defined (i.e., minimum number of simulation steps required to process an activity). Certainly, this duration may be increased according to the time required to retrieve the set of activity-related content needs. An overview of these attributes is presented in Table 5.13.

5.2.3.3 Underlay Network Parameters

Clustering and Topology. The underlay network used within the proposed simulation model organises nodes in clusters. This reflects the typical spatial clustering of smart products known from most envisaged application scenarios. Each cluster consists of one router that enables intra- as well as inter-cluster communication and is assigned a configurable upper bound on the number of participating nodes. To enable inter-cluster communication given node churn, inter-cluster topology is realised as a scale-

Simulation Parameter	Distribution Function	Comment
Node storage capacity	Finite-range discrete distribution	Simplified modeling of storage unit as Integer values
Content size	Finite-range discrete distribution	Simplified modeling of storage unit as Integer values
Content access popularity	Zipf-like distribution	Zipf exponent set to a value 0.5 < exponent < 1
Activity duration	Fix minimum activity duration	Minimum number of simulation steps required to process workflow-related activities; duration can be increased it it takes longer to retrieve activity-related content needs

Table 5.13: Simulation Parameter: Node, Content, Workflow

free network using the topology generation model introduced by Barabási and Albert [BA99]. Underlay nodes within a cluster are arranged in a star topology, with the router representing the central node. While clusters are randomly placed on a virtual map, intra-cluster distribution of underlay nodes follows a heavy-tailed distribution.

Communication. Inter-cluster communication passes at least two routers, namely the router of the cluster the source belongs to and the one of the cluster the destination is associated with. Communication between two underlay nodes of the same cluster is always routed via the local router.

Indeed, communication in smart product systems may also be based on ad-hoc connections. For example, this way of communication can be assumed for smart products being equipped with short-range communication technologies, only. With regard to the distributed storage and retrieval of content in smart product systems, the proposed simulation model assumes nodes to be organised in P2P overlay networks. Smart products being able to establish ad-hoc connections, only, are assumed to make use of nodes in the overlay network to proxy their content storage and retrieval requests. These additional requests are incorporated in the modelling of evaluation scenarios by increasing the request rates of nodes in the P2P overlay network. An example for this simplified modelling approach is presented in Section 5.2.5.

Delay and Bandwidth Modelling. According to the main purpose of the underlay network layer, it only makes use of static delay and bandwidth metrics, which are created prior to simulation execution. This way, the delay and the minimal available bandwidth of a path between any two nodes are statically available. While this reduces simulation accuracy, it clearly enhances simulation execution performance. Delivery time of messages sent in the underlay network is affected by link delays and the smallest bandwidth

of the path between sender and receiver. While the former mainly affects delivery time of small messages (e.g., *GET* request message), the latter has an impact on messages with large content objects (e.g., *GET* response message).

Communication Failure Handling. The underlay network requires at least as much underlay nodes as the maximum amount of nodes that may join the network during simulation. Note that instead of keeping idle underlay nodes to enable proper communication failure handling in case of node *FAIL* and *LEAVE* events, the simulation model maintains a mapping between overlay nodes and underlay nodes, and includes explicit failure handling for expected mappings not being available. This reduces the overall number of underlay nodes required to model a given evaluation scenario.

5.2.4 Quality Attributes

This section classifies and describes the quality attributes used to assess the proposed content placement strategies. This includes network properties covering basic information about nodes and content in the network as well as replication properties that comprise – amongst others – (transient) replica ratios and the utilisation of transient replicas. Moreover, the impact of the proposed content placement strategies on query efficiency and content availability is addressed in detail on different levels of granularity. The scalability of the proposed content placement strategies measured in terms of computation and message complexity is assessed individually without any common quality metric.[42]

Altogether, the defined quality attributes enable comparison of the proposed content placement strategies with related work as well as detailed analyses of the performance of the proposed strategies using quality attributes that are applicable to the latter, only. An overview of the defined quality attributes is presented in Table 5.14 and explained in the following sections.

5.2.4.1 Network Properties

QA1 and QA2. The quality attribute *QA1* measures the average number of *nodes in the network* as well as the evolution of the number of nodes in the network over time given node churn caused by the overlay network events described in Section 5.2.2. This measurement is further applied to the *content stored in the network* for which content, replicas, and transient replicas are distinguished (*QA2*). Both *QA1* and *QA2* are illustrated by means of line charts with simulation steps and accumulated number of nodes / content being used as values for the x-axis and the y-axis, respectively.

[42] Note that – unless specified differently – all average values used in the evaluation study are calculated as arithmetic mean values.

Quality Attribute	Description
QA1	Nodes in network
QA2	Content / replica / transient replica in network
QA3	Content popularity (aggregated)
QA4	Replica ratio
QA5	Transient replica ratio
QA6	Transient replica utilisation
QA7	Query efficiency (overall)
QA8	Query efficiency (workflow)
QA9	Query efficiency (content class)
QA10	Relative query efficiency (overall)
QA11	Relative query efficiency (workflow)
QA12	Relative query efficiency (content class)
QA13	Content availability (overall)
QA14	Content availability (content class)
QA15	Relative content availability (overall)
QA16	Relative content availability (content class)

Table 5.14: Overview of Quality Attributes

QA3. *QA3* covers the overall *content popularity* measured by the number of requests per content regardless of its type (i.e., aggregated view on content, replicas, and transient replicas). This data is plotted using a line chart with the y-axis showing the number of requests and the x-axis showing content objects sorted in descending order of the number of requests. According to the parameter specification presented in Section 5.2.3, content popularity should reflect a Zipf-like power-law distribution.

5.2.4.2 Replication Properties

QA4 and QA5. In general, query efficiency and content availability can be enhanced up to a certain level by distributing a high number of replicas in the network, i.e., by high replication degrees. However, given the limited resources of smart products, content replication strategies for smart product systems have to address the trade-off between enhancement of query efficiency and content availability on the one hand, and the replication degree required for achieving these objectives, on the other hand. This is covered by the quality attributes *QA4* and *QA5*, which capture the *ratio of replicas and transient replicas*, respectively. Both are formulated as the ratio of the average number of (transient) replicas in the network and the average number of content in the network.

QA6. In order to assess the proactive nature of the proposed replication strategies and the related risk of false pre-replication, the *utilisation of transient replicas* is assessed by quality attribute *QA6* in terms of the number of accesses. Hence, *QA6* covers values in the interval $[0, n-1]$ with n representing the maximum number of accesses required for transforming a replica from transient to persistent state. A value of 0, on the other hand, indicates either *false pre-replication* if the transient replica was created by one of the workflow-based replication strategies or *inefficient placement* if the transient replica was created by the content-class-based replication strategy. Note that the quality attribute measures the maximum number of requests, i.e., the highest TTL interval achieved, only. *QA6* is illustrated based on pie charts showing the portion of transient replicas with $0, 1, ..., n-1$ accesses.

5.2.4.3 Query Efficiency Properties

QA7, QA8, and QA9. The impact of the proposed replication strategies on query efficiency is measured by multiple quality attributes. First, *QA7* is used to analyse the overall query efficiency measured in terms of the RTT of *GET* requests. Second, *QA8* focuses on the effect of the proposed workflow-based replication strategies by measuring query efficiency of *GET* requests associated with activity-related content needs in the control flow of workflows, only. Third, *QA9* enables assessment of the impact of the proposed content-class-based replication strategy on query efficiency by solely taking into account *GET* requests for content with class *CC_QUERY_EFFICIENCY* being assigned by the requester.

QA10. In order to analyse the impact of replication degree on query efficiency, the quality attribute *QA10* weights the overall query efficiency (*QA7*) by the number of (transient) replicas of a requested content. The assessment of this *relative query efficiency* is done based on the formula presented in Eq. 12. The overall query efficiency referred to as *RTT(GET request)* is multiplied with a weighting factor $\omega(id)$ with id representing the identifier of the requested content (see Eq. 13). This factor captures the number of replicas and transient replicas of the requested content. Replicas and transient replicas are assessed separately and weighted by the penalisation factor α, which is set to a value in the interval $[0, 0.5)$. This leads to transient replicas being assigned stronger weighting, which is reasonable since they are typically created to enhance query efficiency. Note that in the analysis of the simulation results presented in Section 5.4.2, the penalisation factor is set to a value of $\alpha = 0.4$.

To weaken the actual impact of replicas and transient replicas, their replication degree is measured in terms of the natural logarithm. Moreover, replication degree is increased by a value of 1. This way, a (transient) replication degree of 0 results in a value of $log(1) = 0$ and does not affect the relative query efficiency property. Finally, the two weighted replication degrees are summed up and increased by a value of 1 to ensure that ω has no impact on query efficiency in case of no (transient) replicas being associated

$$QA10 = RTT(GET\ request) \times \omega(id) \tag{12}$$

$$\omega(id) = 1 + \alpha \times log(1 + |R_{id}|) + (1 - \alpha) \times log(1 + |T_{id}|) \tag{13}$$

R_{id} : set of replicas of content with identifier id
T_{id} : set of transient replicas of content with identifier id
α : penalisation factor, $0 \leq \alpha < 0.5$

with the requested content. Hence, the weighting factor ω presented in Eq. 13 increases, i.e., worsens, query efficiency with the number of (transient) replicas, with transient replicas having a stronger impact than replicas.

QA11 and QA12. This concept of relative query efficiency is further applied to the quality attributes *QA8* and *QA9* to enable detailed evaluation of the proposed content placement strategies. The results are captured by the quality attributes *QA11* and *QA12* that relate to *QA8* and *QA9*, respectively.

All (relative) query efficiency attributes (*QA7* - *QA12*) are illustrated by means of line charts with the x-axis showing simulation steps and the y-axis showing the average (relative) query efficiency of *GET* requests made at a certain simulation step.

Note that for all query efficiency quality attributes, *GET* requests triggered within the content replication phase of the workflow-based replication strategies are not taken into consideration. These requests are classified as "passive" requests, because they are meant for enhancing query efficiency of the actual *GET* requests. Including them into the analysis would distort the actual impact of the strategies on query efficiency.

5.2.4.4 Content Availability Properties

QA13 and QA14. Content availability is measured in terms of the hit rate, i.e., the portion of *GET* requests for which a *GET* response is received. The related quality attribute *QA13* is illustrated based on a line chart with simulation steps on the x-axis and the hit rate of all *GET* requests dispatched at a certain simulation step on the y-axis. Moreover, similar to the query efficiency quality attribute *QA9*, the quality attribute *QA14* enables detailed analysis of the impact of the replication strategy CC by only taking into account *GET* requests for content with class *CC_AVAILABILITY* being assigned by the requester.

QA15 and QA16. Moreover, *QA15* assesses content availability by taking into account the replication degree of (transient) replicas related to the requested content (as opposed to the simple available / unavailable metric applied by *QA13* and *QA14*). As presented

$$QA15(t) = \frac{1}{|\{GET\ requests\ dispatched\ at\ t\}|} \times \sum_{\forall GET\ requests\ dispatched\ at\ t} f(id) \quad (14)$$

$$f(id) = \begin{cases} \frac{1}{\omega(id)} & \text{if content is available} \\ \beta \times \omega(id) & \text{if content is not available and } R_{id} \cup T_{id} \neq \emptyset \\ 0 & \text{if content is not available and } R_{id} \cup T_{id} = \emptyset \end{cases} \quad (15)$$

R_{id} : set of replicas of content with identifier id

T_{id} : set of transient replicas of content with identifier id

α : penalisation factor, $0.5 < \alpha \leq 1$

β : weighting factor, $\beta < 0$

in Eq. 14, each *GET* request is assessed by the metric $f(id)$ depicted in Eq. 15 with id representing the identifier of the requested content. This metric differentiates three cases depending on the actual content availability and the (transient) replication degree related to the requested content.

- **Content is available.** If the requested content is available, then the multiplicative inverse of the weighting factor $\omega(id)$ is applied (see Eq. 13). Note that the application of the multiplicative inverse is required, because relative content availability worsens with decreasing values whereas relative query efficiency worsens with increasing values.
- **Content is not available and $R_{id} \cup T_{id} \neq \emptyset$.** If content is not available in the presence of (transient) replicas related to the requested content, then the non-availability is penalised by the factor $\beta \times \omega(id)$ with $\beta < 0$. Note that this requires again multiplication with the compound natural logarithm expression, because the negative impact of unserved requests should increase with an increasing number of (transient) replicas. This enables penalisation of non-optimised content placement, because – obviously – content placement strategies do not perform properly if content is unavailable even though there are (transient) replicas distributed in the network. Note that for both cases the penalisation factor α (see Eq. 13) is adapted compared to the relative query efficiency quality attributes in order to account for the stronger impact of replicas on content availability. In the analysis of the simulation results presented in Section 5.4.2, the parameters α and β are set to values of 0.6 and -0.5, respectively.
- **Content is not available and $R_{id} \cup T_{id} = \emptyset$.** Finally, if content is not available and there are no (transient) replicas distributed in the network, then there is no penalisation and the non-availability is simply assessed with a value of 0.

Consequently, the applied metric weakens content availability and penalises its non-availability with the number of related (transient) replicas. This enables detailed analysis of the extent to which the proposed strategies optimise content placement. The metric is applied to the two quality attributes *QA13* and *QA14* resulting in the relative content availability attributes *QA15* and *QA16*, respectively.

5.2.5 Evaluation Scenario

Basic Assumptions. In order to enable evaluation of different content placement strategies, the smart aircraft manufacturing scenario described in Section 2.2.3 was modelled in cooperation with EADS Innovation Works in the course of the SmartProducts project.[43] The model simulates half a shift, i.e., 4 hours, in a single plant.[44] This plant is assumed to consist of 10 workstations with a maximum number of 100 blue-collar workers per workstation. Each worker is assumed to carry 1 nomadic device as well as up to 3 smart tools. To simplify simulation modelling, workstations are assumed to be used to assemble one smart aircraft component at the same time (sequential production). Non-smart aircraft components are not explicitly modelled as they are assumed to not being capable of participating in the overlay network. Instead, they use other smart products as proxies for *PUT / GET* requests. These requests are indirectly considered by adapting the request rate of the associated smart aircraft components. Finally, it is assumed that the assembly of smart aircraft components lasts 2 hours on average; the storage unit used to model storage capacity of smart products as well as content size represents 0.1 MB.

In order to determine the scale of one simulation step, multiple simulation runs with an average content size of 10 storage units (1 MB) and different configurations were performed and analysed. This analysis resulted in an average delivery time of 10 simulation steps for a *GET RESPONSE* message. Based on the assumption of an average delivery time of 5 seconds for a message with a payload of 1 MB, a simulation step is scaled to reflect 0.5 seconds. Consequently, the simulation of 4 hours lasts 28,800 simulation steps.

5.2.5.1 Modelling of Node Classes

Workstations. Workstations are modelled as stable nodes that join during the simulation initialisation phase; they neither fail nor leave the network. Workstations are assumed to possess an average storage capacity of 255 GB.

[43] Since there is no prototype of the presented scenario available, the configuration of most parameters is based on assumptions that reflect the state of work when writing this thesis.

[44] Limiting the scope to a single plant is valid, because the focus of the simulation is on the extent to which smart products enhance performance of aircraft manufacturing processes.

Nomadic Devices. Based on the assumption of an average number of 90 blue-collar workers per workstation with each worker carrying 1 nomadic device, the simulation model consists of 900 nomadic devices on average. Nomadic devices are assumed to possess on-board storage with an average storage capacity of 5,400 MB and to have an average lifetime (in terms of network participation) of 4 hours, which reflects a single break per shift.

Aircraft Components. Due to the assumption of sequential production, the simulation model includes 10 aircraft components on average (one per workstation). As described above, aircraft components participate in the network for an average period of 2 hours. They are assumed to have an average storage capacity of 32.5 MB.

Smart Tools. Based on the assumption of each blue-collar worker carrying 2 smart tools on average, there is an average number of 1,800 smart tools with an assumed average storage capacity of 1.6 MB. Due to their limited resources, smart tools are assumed to have an average lifetime of 30 minutes. This reflects the fact that smart tools tend to deactivate communication in order to save energy in case they are not actively used (e.g., after completion of a process they have been used for).

5.2.5.2 Modelling of Workflows

Since all activities in the evaluation scenario are based on well-defined workflows, the determination of *PUT / GET* request rates per node class is derived from the workflow execution configuration. It is assumed that workflows consist of 6 activities of which each requires the usage of 2 smart tools (average values). The average number of content objects needed per activity is assumed to approximate a value of 2.37. Moreover, workflows are assumed to be executed every 30 minutes on average. To take into consideration idle time between processing of activities and workflows (e.g., for changing equipment required for task accomplishment), activity execution is assumed to last half a minute on average. Workflows are solely started by nomadic devices and each nomadic device is only capable of starting a single workflow at the same time. It is assumed that each blue-collar worker performs 3 workflows during the assembly of an aircraft component. While smart tools store results of each activity, nomadic devices, workstations, as well as aircraft components only store aggregated results of completed workflows.

To simplify the scenario, the 4 different workflow types presented in Figure 5.17 are defined. Details about the assignment of workflows to nomadic devices on the one hand as well as of content needs to activities on the other hand are presented in Sections 5.4.1 and 5.3.5, respectively.

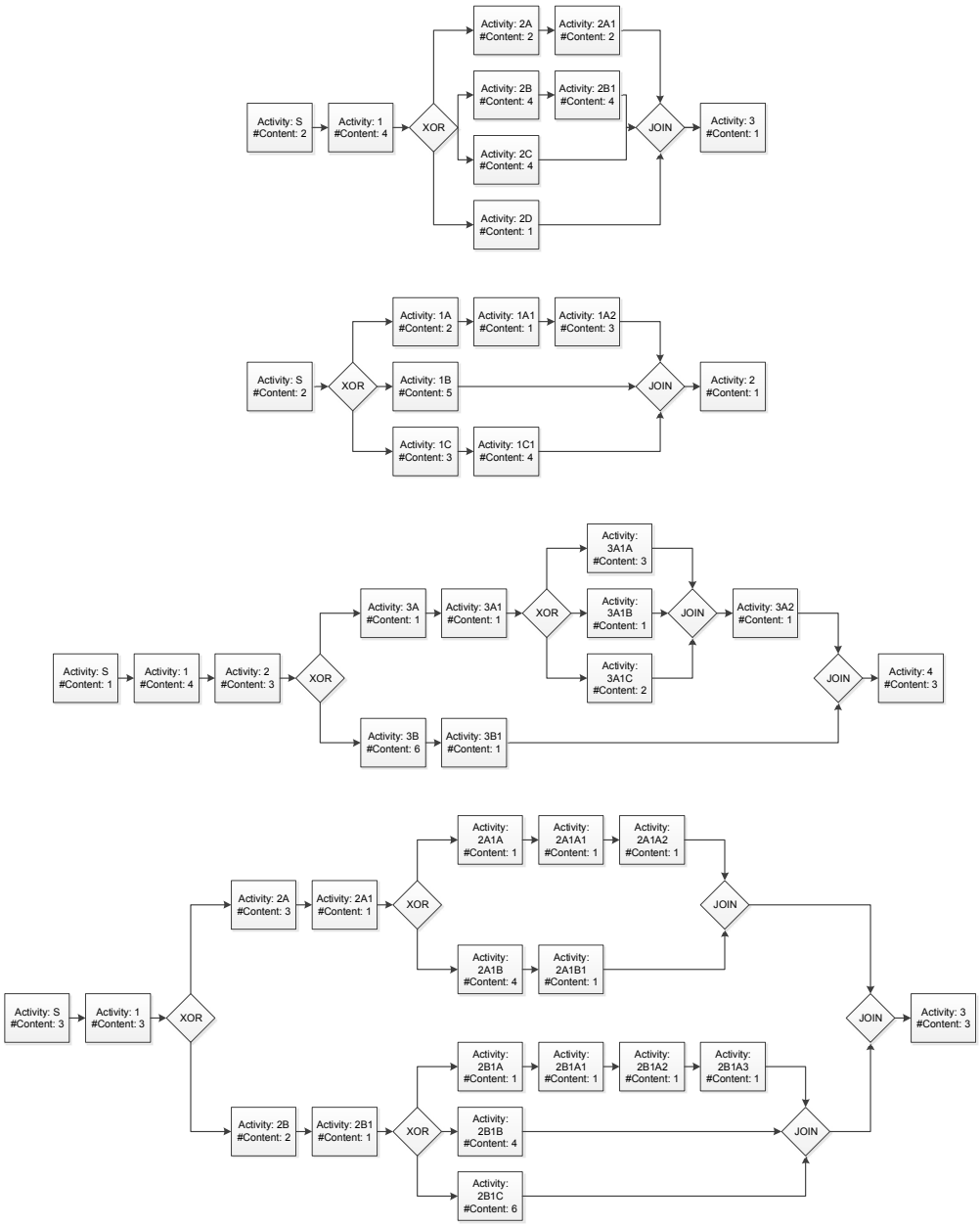


Figure 5.17: Evaluation Scenario: Modelling of Workflows

5.2.5.3 Modelling of PUT / GET Request Rates

Smart Tools. Smart tools have an average lifetime of roughly one workflow duration. Moreover, since each blue-collar worker is carrying 2 smart tools and 2 smart tools are required for each activity on average, each smart tool is used in 6 activities on average during its lifetime. Consequently, smart tools submit 6 *GET* requests (request of activity-related configuration) and 6 *PUT* requests (storage of activity result) on average.

Nomadic Devices. Nomadic devices are alive for approximately the assembly of 2 aircraft components and are used to execute 3 workflows per aircraft component. Hence, since they store results of executed workflows they have an average *PUT* request rate of 6 (2×3 workflows). The composition of the *GET* request rate of nomadic devices is more complex. First, in order to aggregate overall workflow execution results, they request operation results from smart tools, which deliver data in aggregated form. Consequently, since 2 smart tools are used on average per workflow, this results in a number of 12 *GET* request (2×3 *workflows* $\times 2$ *smart tools*). In addition, workflows to be performed are requested from aircraft components / workstations. According to their lifetime, this leads to 6 *GET* requests (2×3 *workflows*). Finally, based on the assumption of each activity requiring 2.37 different content objects on average, an additional number of 85 *GET* requests is dispatched (2×3 *workflows* $\times 6$ *activities* $\times 2.37$ *content objects*). Hence, within their lifetime, nomadic devices have an average *GET* request rate of 103 ($6 + 12 + 85$ *GET* requests).

Aircraft Components. Aircraft components store an aggregated view on the results of all executed workflows. Due to an average number of 90 blue-collar workers per workstation with each worker performing 3 workflows to assemble an aircraft component, this leads to 270 *GET* requests (3×90 *workflow results*)) and a single *PUT* request. Moreover, in order to account for multiple versions of the aggregated view, the *PUT* request rate is increased to the value of the *GET* request rate. As a side effect, this also increases dynamics during simulation execution by additional content being injected in the simulated network.

Workstations. Finally, workstations request the aggregated view of the associated aircraft components and store a compound view in order to obtain digital representations of aircrafts as being manufactured. This leads to 2 *GET* requests given that 2 aircraft components are being assembled in the given timeframe of 4 hours and 1 *PUT* requests. Again, following the argumentation of the increased *PUT* request rate of aircraft components as well as to cover *PUT* requests for storing workflows to be accomplished, the *GET* / *PUT* request rates are increased to a value of 540.

Simulation Parameter	Workstation		Aircraft Component		Nomadic Device		Smart Tool	
Number of nodes (avg.)	10		10		900		1,800	
Storage capacity [portion]	1,500,000	[0.3]	250	[0.7]	20,000	[0.2]	10	[0.7]
	3,000,000	[0.7]	500	[0.3]	50,000	[0.6]	20	[0.2]
					100,000	[0.2]	50	[0.1]
Node *JOIN*	Join in simulation initialisation phase only		Poisson(14,400)		Poisson(28,800)		Poisson(3,600)	
Node *FAIL*	Stable		Weibull(10, 14,400) P(*FAIL*) = 0.05		Weibull(15, 28,800) P(*FAIL*) = 0.02		Weibull(6, 3,600) P(*FAIL*) = 0.1	
Node *LEAVE*	Stable		Weibull(10, 14,400) P(*LEAVE*) = 0.95		Weibull(15, 28,800) P(*LEAVE*) = 0.98		Weibull(6, 3,600) P(*LEAVE*) = 0.9	
PUT request	Poisson(53.3)		Poisson(52.6)		Poisson(4,500)		Poisson(500)	
GET request	Poisson(53.3)		Poisson(52.6)		Poisson(1,550)		Poisson(500)	
WORKFLOW execution	--		--		|Normal(3,600, 300)|		--	
Activity duration	--		--		60		--	
Content type	A		B		A		B	

Table 5.15: Evaluation Scenario: Modelling of Simulation Events and Node Classes

Simulation Parameter	Content Type A		Content Type B	
Content size [portion]	10	[0.4]	2	[0.8]
	20	[0.3]	5	[0.2]
	50	[0.2]		
	100	[0.1]		
Content access popularity	Zipf-exponent	0.8	Zipf-exponent	0.8
Content class [portion]	*CC_DEFAULT*	[0.2]	*CC_DEFAULT*	[0.2]
	CC_AVAILABILITY	[0.5]	*CC_AVAILABILITY*	[0.5]
	CC_QUERY_EFFICIENCY	[0.3]	*CC_QUERY_EFFICIENCY*	[0.3]

Table 5.16: Evaluation Scenario: Modelling of Content Parameters

The detailed modelling of node classes and simulation events, as well as content types is presented in Tables 5.15 and 5.16, respectively. This configuration results in a theoretical ratio of activity-related *GET* requests to non-activity-related *GET* requests of approximately 45%.[45]

[45] Given the event configuration presented in Table 5.15, the ratio is calculated as follows. The workflows used in the evaluation scenario have an average path length of 6 activities that each require 2.37 content objects on average. With an average number of 900 nomadic devices being able to execute workflows and an average number of 8 workflow executions per nomadic device during the defined simulation time, this yields a value of 102,384 (exact value given the assumed workflows equals 102,418) activity-related content requests. The expected number of non-activity-related content requests is calculated for each node class by multiplying the average number of nodes with their above-explained *GET* request rate. Summed up over all node classes and taking into account the lifetime per node class, this leads to a total number of 125,327 non-activity-related content requests and reveals the above-stated ratio of 45%.

5.3 Implemented Simulation Model

The extended overlay simulation model is implemented based on the open-source overlay simulation engine PlanetSim[46]. This simulation engine is written in the object-oriented programming language Java and complies with the Common API to clearly divide overlay network and overlay service functionality. PlanetSim requires an implementation of the functionality of the overlay network layer according to the Application Programming Interface (API) and concepts of the simulation engine. Yet, all functionality layered above the Common API can be simulated without any need for adaptation.

While PlanetSim is known to enable efficient simulation of P2P content location and routing substrates, it relies on virtual random topologies, only [ALA+08]. For the purpose of enabling accurate simulation of the proposed content placement strategies, the simulation engine has been extended by an underlay network layer that enables integration of realistic topologies generated by the network topology generator BRITE [MLMB01].

This section presents the implementation of the extended overlay simulation model. This includes high-level design descriptions covering both the compositional and dynamic structure of the implemented simulation model, and detailed design descriptions of the main components of the underlay network, overlay network, and overlay service layer. Whenever beneficial, these descriptions are complemented by design diagrams such as Fundamental Modelling Concepts (FMC) block diagrams, Unified Modelling Language (UML) state charts, and UML class diagrams.

5.3.1 High-Level Design Description

5.3.1.1 Compositional Structure of the Implemented Simulation Model

Simulation Engine and Reporting Engine. The implemented simulation model presented in Figure 5.18 reflects the logical layers of the conceptual simulation model (see Section 5.2.1). The core component is the Simulation Engine, which orchestrates the components of the aforementioned layers and controls the simulation process. The simulation engine reads in both the evaluation scenario configuration and the overall simulation configuration, which covers – amongst others – message-related data such as maximum number of hops and timeout properties as well as the configuration of the proposed content placement strategies. Also, the simulation engine is responsible for drawing samples of the distribution functions used by the simulation model as well as

[46] http://projects-deim.urv.cat/trac/planetsim/

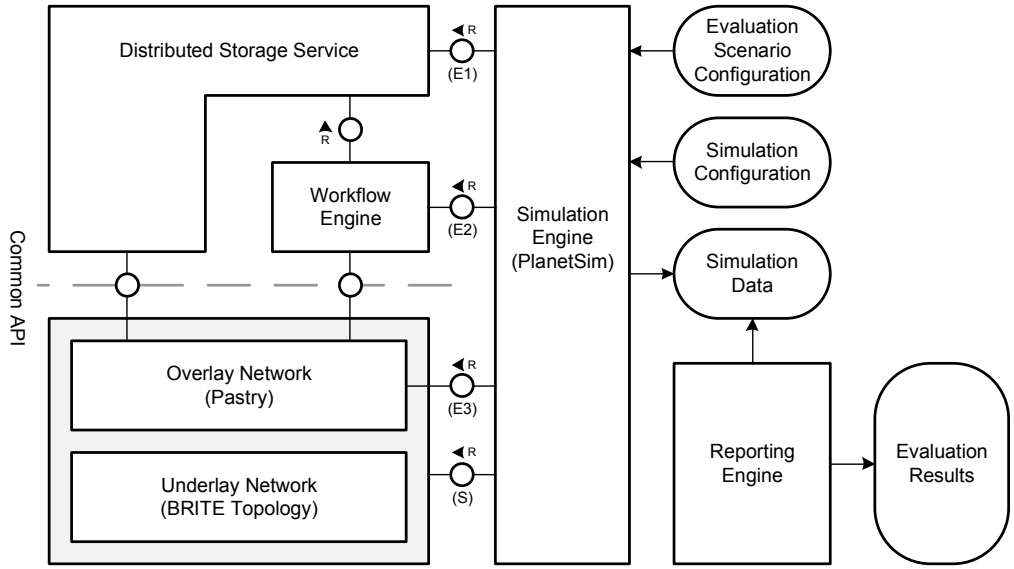


Figure 5.18: Compositional Structure of the Implemented Simulation Model

for the collection of raw simulation data. At the end of the simulation execution, this data is processed by the Reporting Engine to derive and compute the simulation results and generate simulation reports given the quality attributes of the simulation model.

As depicted in Figure 5.18, the simulation engine has distinct interfaces to the components of the following three layers to control the simulation process.

Overlay Service Layer. The overlay service layer consists of the Distributed Storage Service that provides P2P-typical content placement and retrieval functionality. This service enables plugging-in the proposed content placement strategies and provides the functional basis for the latter. Also, the overlay service layer covers a P2P Workflow Engine, which is required to simulate the proposed workflow-based replication strategies. The simulation engine controls both components by means of *PUT / GET* events and *WORKFLOW* events via the interfaces *E1* and *E2*, respectively.

Overlay Network Layer. The overlay network layer includes an implementation of a P2P content location and routing substrate. According to [Mü07b], hybrid structured overlay networks are most suitable for smart products systems as they reflect their inherent heterogeneity. Yet, in order to reduce complexity, the simulation model incorporates a plain structured overlay network implementation. This enables operation and evaluation of the proposed content placement strategies and demonstrates their applicability in diverse setups. Moreover, as discussed in Section 4.4, the impact of the proposed strategies is assumed to be even stronger in hybrid structured overlay network configurations.

Due to the location-characteristics of the proposed content placement strategies, the implemented simulation model provides an implementation of the overlay network *Pastry* [RD01a] that makes use of the Common API to facilitate data exchange with components of the overlay service layer. This component is managed via interface *E3* by means of the overlay network events presented in Figure 5.16.

Underlay Network Layer. The underlay network layer enables simulation with realistic topologies. Since both overlay and underlay network layer are implemented according to the API of the simulation engine, there is no direct data / message exchange between them. Instead, it is the simulation engine that controls behaviour and exchange of the two layers via the interface *S*.

5.3.1.2 Dynamic Structure of the Implemented Simulation Model

The dynamic structure of the simulation model is best illustrated by means of the interaction between the main components of the simulation model during simulation execution. The following description bases on the node-centric view of the simulation model depicted in Figure 5.19.

The simulation engine maintains a set of nodes that belong to the overlay network. In the implemented simulation model, these nodes are realised as Pastry nodes, i.e., nodes that are capable of building up a structured overlay network according to the P2P content location and routing substrate Pastry. Each of these nodes is assigned an appropriate node in the underlay network layer – a so called BRITE node – that reflects the node class of the former. Moreover, each Pastry node has its own instance of the two overlay services distributed storage service and workflow engine.

The communication between the overlay network and overlay service layers is facilitated by the Common API provided by the simulation engine. Message transmission is fully simulated by the simulation engine. Hence, message are not transmitted but handed over from one node (source) to another (intermediate / destination) by means of delayed message re-assignment given the delivery times derived from the BRITE topology of the underlay network layer.

The core simulation process consist of 4 main phases: simulation task processing, simulation event processing, message delivery, message processing. These phases are processed for each simulation step until the simulation end event scheduled in the simulation initialisation phase is reached.

Phase 1: Simulation Task Processing. This phase covers processing of all simulation tasks scheduled for the given simulation step. Simulation tasks enable realisation of timer tasks known from Java development in the course of simulation execution. Hence, object-related tasks can be planned for certain simulation steps and can be re-scheduled periodically, if needed. For example, the timeout handling of *GET* requests or the control

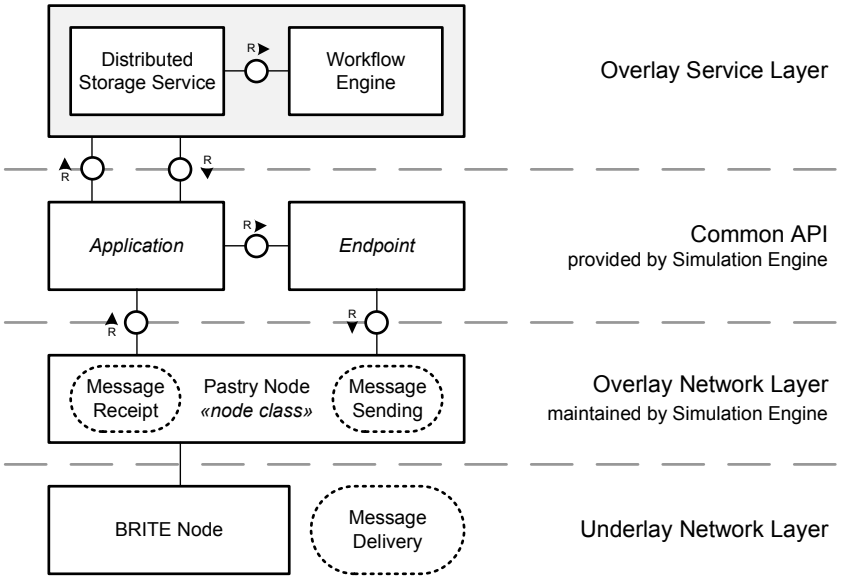


Figure 5.19: Implemented Simulation Model: Node-centric View

of the gradually increasing TTL property of transient replicas are both controlled by simulation tasks. This way, simulation tasks complement the use of simulation events, which may be classified as external triggers.

Phase 2. Simulation Event Processing. After processing of pending simulation tasks, all events scheduled for the given simulation step are handled according to the overall simulation control flow and the simulation event graph presented in Figures 5.14 and 5.16, respectively. For example, the *processing phase* of the *JOIN* event creates a new Pastry node including the related overlay service stack. This node is assigned a node class as well as a storage capacity given the range configured for the node class. The actual determination of node classes is done dynamically during the simulation initialisation phase taking into consideration scenario-specific distribution of node classes. Thereafter, in the simulation execution phase, node class assignment is pre-defined according to the node-centric event scheduling approach. The new node is then (i) added to the network maintained by the simulation engine and (ii) associated with a BRITE node in the underlay network that fits its node class. Finally, the new node joins the overlay network via a random bootstrap node.

In the *scheduling phase* of the event, new events are scheduled according to the simulation event graph (node class is also used for new join events). Note that overlay service events are scheduled with a pre-defined delay to account for overlay stabilisation. An overview of the two phases of the join event is presented in Figure 5.20. The process of the remaining overlay network and service events is outlined in Annex A.1.

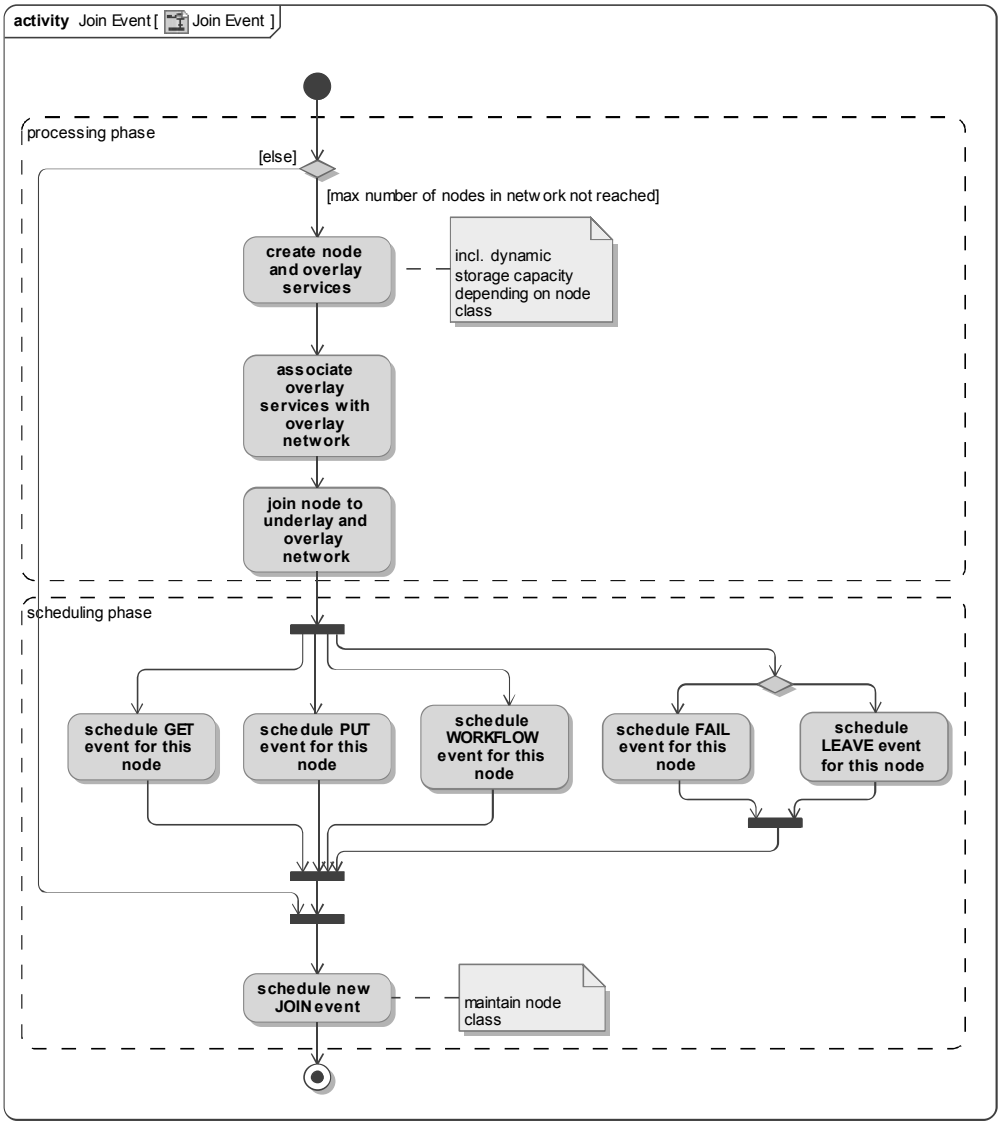


Figure 5.20: Process Flow of JOIN Events

Phase 3: Message Delivery. The next phase covers message delivery within the underlay network. For this purpose, the underlay network layer maintains the map Message Delivery that associates all messages to be delivered with their delivery times represented as the simulation step at which the message is handed over to its recipient (intermediate or destination). Message delivery is realised by the underlay network invoking the receive method of the recipient – a Pastry node – which stores the message within its Message Receipt collection for further processing.

Phase 4: Message Processing. After delivery of all messages with delivery time matching the given simulation step, messages are processed by Pastry nodes (i.e., the recipients). Certainly, processing of messages can lead to the creation of new messages that are to be sent / routed to their destination. For example, a *GET* request processed by the distributed storage service may lead to a *GET RESPONSE* message to be sent back to the requester. Pastry nodes maintain these message in the Message Sending collection. After having processed all messages, the Message Sending collections of all nodes are being handled. This includes determination of the message delivery time given the recipient and the delivery path defined by the BRITE topology. Eventually, this results in putting these messages in the Message Delivery map of the underlay network layer according to their calculated delivery times.

5.3.2 Underlay Network

5.3.2.1 Compositional Structure

The compositional structure of the underlay network layer is presented in Figure 5.21. The two core components of the layer are (i) the Underlay Network, which enables scheduling and delivery of messages including failure handling, and (ii) the BRITE Network, which establishes the association between Pastry nodes and BRITE nodes, and which provides the operational basis of the former. Moreover, the BRITE Network encapsulates specifics of the BRITE topology and enables calculation of delay and bandwidth between any two BRITE nodes based on static metrics. While the BRITE Network is tailored to the usage of BRITE topologies, the Underlay Network decorates the former and provides a generic interface to be used in the simulation environment.

The functionality of the BRITE Toolset presented in Figure 5.21 is used to prepare the static metrics (BRITE Nodes, Routes, Bandwidth Matrix, Delay Matrix) used in the underlay network layer during simulation execution. The toolset includes the BRITE GUI of the BRITE topology generator, which enables configuration and generation of BRITE topologies. Moreover, the BRITE Adapter has been implemented to parse and adapt BRITE topologies according to the specifics of smart product systems described in Section

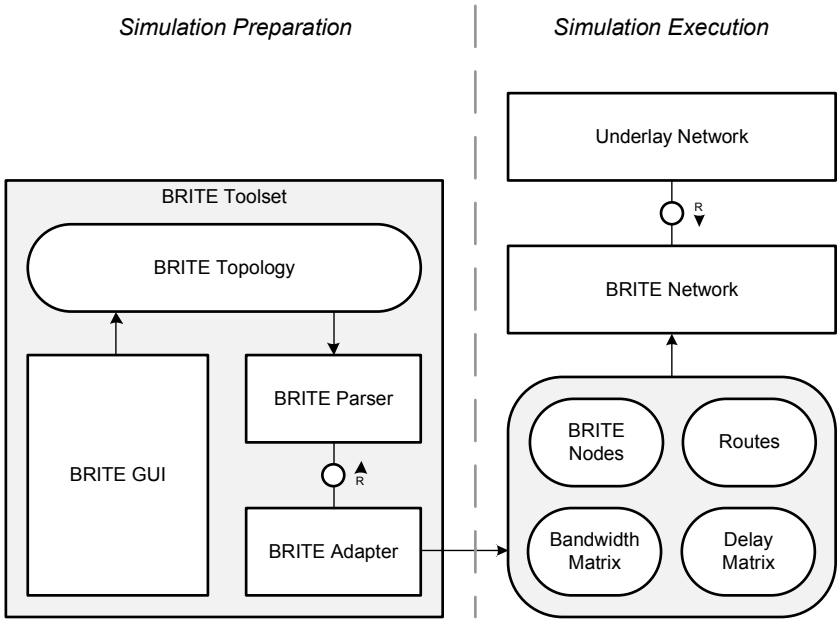


Figure 5.21: Implemented Simulation Model: Compositional Structure of the Underlay Network

5.2.3, as well as to persist the metrics of the adapted topologies for enhancing simulation execution performance. These static metrics are used during simulation execution by the BRITE Network in order to avoid recurring calculation as well as time-consuming on-demand determination of shortest paths between BRITE nodes.

The following sections detail the two steps applied within the simulation preparation phase to create and persist the above-mentioned static metrics, namely the network topology generation step (Section 5.3.2.2) and the network topology adaptation step (Section 5.3.2.3). Moreover, the integration of the BRITE topology in the simulation framework is detailed in Section 5.3.2.4.

5.3.2.2 Network Topology Generation

According to the specifics of the underlay network layer modelling presented in Section 5.2.3, the BRITE topology is set up as a 2-level topology with multiple Autonomous Systems (AS). Even though the presented evaluation scenario covers a single AS only, this approach is applied to reflect node clustering that is typical for smart product sys-

tems.[47] By definition, each AS consists of several routers and at least one border-router that enables inter-AS communication. In the resulting topology, each router is connected with other routers and / or border-routers; border-routers are interconnected across ASs.

To ensure a sufficient number of nodes in the topology for the simulation of the evaluation scenario described in Section 5.2.5, the topology consists of 10 ASs with 1000 nodes each. The link bandwidth within ASs is distributed uniformly within the interval [11, 54] Mbits/s; link bandwidth across ASs (i.e., bandwidth of links between border-routers) is set to a constant value of 1 Gbit/s. The assignment of link delays is fully covered by the BRITE topology generator.

The actual configuration set in the BRITE GUI as well as the resulting BRITE topology are illustrated in Annexes A.2.1 and A.2.2, respectively.

5.3.2.3 Network Topology Adaptation

Obviously, the topology generated by BRITE does not fully match the underlay network model defined in the conceptual simulation model. For example, there is no star topology within clusters and each cluster may consist of multiple border-routers. The network topology adaptation approaches this issue and provides the static metrics used during simulation execution.

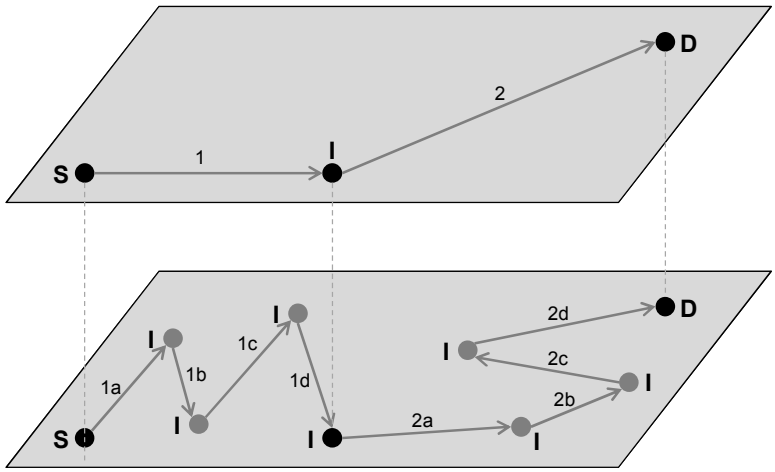
Step 1. Shortest Paths Determination. In a first step, the "shortest" path between any two routers is determined based on Dijkstra's shortest path algorithm [Dij59]. The metric used by the algorithm takes into account both the delay of a path from source to destination calculated as the sum of the delays of the links between any two nodes of the path, and the bandwidth of a path, which is set to the minimum bandwidth of the nodes along the path. The shortest paths are persisted in two two-dimensional nilpotent upper triangular metrics for path delay and bandwidth between any two routers.

Figure 5.22 presents an example for this approach. A message sent from source *S* to destination *D* passes several intermediates along the links *1a – 2d*. The associated path has a delay of 7.9 seconds and a bandwidth of 11 Mbits/s.

Step 2: Topology Adaptation. This step addresses the mismatch between the topology defined in the conceptual simulation model and the one generated by BRITE. This mismatch covers the following three invariants: (i) there is a single border-router per AS, (ii) each router is connected to one border-router, only, and (iii) there is an indirect path between any two routers.

Thus, in each cluster, all border-routers but the one with the highest in- / out-degree are removed. Moreover, for all routers without direct connection to the border-router of

[47] Note that even though other options such as node placement following a heavy-tailed distribution does result in node grouping up to a certain degree [YSL+06], it does not reflect the "clear" clustering of the modelled evaluation scenario.



Edge Id	Delay (sec)	Bandwidth [Mbits/s]
1	4	11
2	3.9	48

Edge Id	Delay (sec)	Bandwidth [Mbits/s]
1a	0.3	54
1b	0.6	48
1c	1.5	54
1d	1.6	11
2a	1.1	48
2b	0.1	54
2c	0.8	54
2d	1.9	54
	7.9	**11**

Figure 5.22: Implemented Simulation Model: Link Delays and Bandwidth within the Underlay Network Topology (example)

their cluster, a new direct connection is added to the topology with the properties of the shortest path between the router and its border-router. Next, all links between routers are removed. This results in a star topologies for all clusters with the border-router playing the role of the central node.

Step 3: Terminology Adaptation. The terminology is adapted to meet the conceptual simulation model. Thus, ASs become clusters, routers become nodes (i.e., BRITE nodes), and border-routers become routers. This results in the multi-cluster topology seen in most smart product scenarios and ensures topologies with realistic properties given that all smart products in the evaluation scenario are organised in a P2P overlay network.

Step 4. Metric Adaptation and Persistence. In the last step, the delay and bandwidth metrics of the shortest paths determined in step 1 are adapted to cover the paths of the adapted topology, only. The adapted metrics as well as collections of nodes and routes are persisted for being used in follow-up simulation executions, and provide the basis for the scheduling of message retrieval described in Section 5.3.1.2.

As an alternative to the proposed approach, a BRITE topology with a single AS could be generated and extended with smart products being linked with routers of the BRITE topology. However, these links would have fictional / random properties. Instead, the proposed adaptation preserves the link properties created with the BRITE topology generator and – hence – can be assumed to enable more accurate simulations. An overview of the adaptation procedure is illustrated by Figure 5.23.

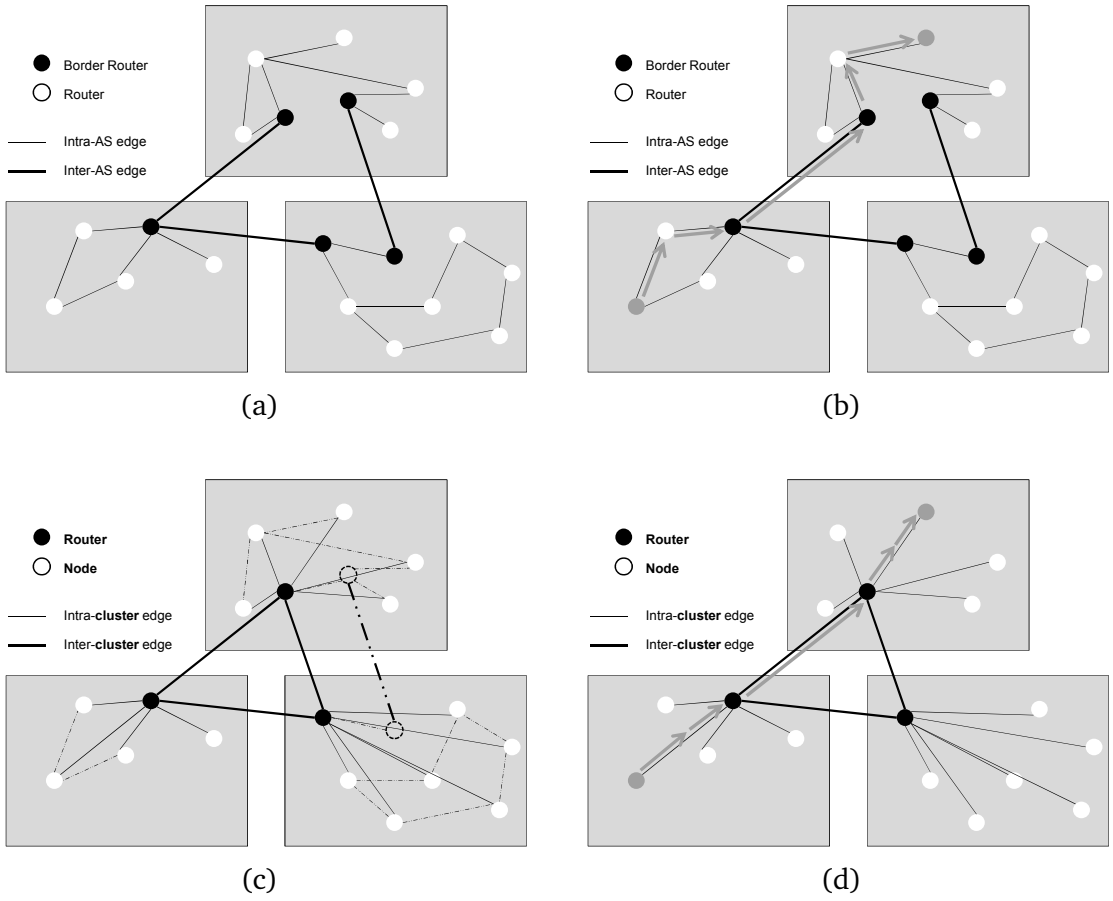


Figure 5.23: Implemented Simulation Model: Underlay Network Topology Adaptation (a) original topology, (b) message delivery in original topology, (c) adapted topology, (d) message delivery in adapted topology

5.3.2.4 Network Topology Integration

Mapping of BRITE Nodes and Pastry Nodes. The heterogeneity of smart products as well as the corresponding modelling of node classes is taken care of in the underlay network layer by means of an appropriate mapping of BRITE nodes and Pastry nodes. The metric used for this mapping covers the bandwidth of BRITE nodes. Hence, all "free" BRITE nodes, i.e., BRITE nodes that are not associated with Pastry nodes, are sorted in descending order of their bandwidth. This sorted list is divided into subsets according to the actual distribution of node classes defined in the evaluation scenario. This way, a joining node of the node class with the highest resources is assigned a BRITE node of the first subset. The actual selection of BRITE nodes within the defined subsets is done randomly. Similarly, nodes leaving the network caused by either *FAIL* or *LEAVE* events result in the mapping being revoked.

Failure Handling. Taking into consideration that the BRITE network does not maintain idle nodes, a specific failure handling is defined that distinguishes the following cases.[48] Note that failure handling is done during message delivery as described in Section 5.3.1.2.

- **Case 1.** When the recipient of a sent or forwarded message left / failed prior to message delivery, a special *FAIL* message encapsulating the original message is created and returned to the last hop. The delivery time of this message is set according to the delay between sender and recipient.
- **Case 2.** When the recipient of a routed message left / failed prior to message delivery, the last hop is notified to enable the overlay network to re-route the message.
- **Case 3.** Whenever a sender left / failed prior to message delivery, the message is discarded regardless of whether the message was sent, forwarded, or routed. Given the purpose of the proposed simulation model, this approach is feasible. If a requester does not receive a response message, it can be inferred that the request failed also if no explicit error message was received.

5.3.3 Overlay Network

5.3.3.1 Pastry Selection

The implemented simulation model incorporates an implementation of the overlay network Pastry (see Sections 5.3.1.1 and 2.1.4). Pastry applies Plaxton's prefix matching approach and enables proximity-aware routing by means of specific proximity metrics.

[48] The complementary failure handling by the overlay network implementation and the distributed storage service are described in Sections 5.3.3.2 and 5.3.4.8, respectively.

Content is commonly placed either on the root node, i.e., the node whose identifier equals the content identifier, or on the next-best assignable node with an identifier succeeding the content identifier in the logical identifier space of the overlay network, i.e., the surrogate root node.

Most important, Pastry maintains a state table – the neighbour set – that includes the nodes "closest" to the local node given the above-mentioned proximity metric. This enables content placement strategies to make use of resources in the environment of the local node for enhancing query efficiency. Moreover, lookup mechanisms can be defined that query neighbour nodes in parallel to the common routing of *GET* requests in order to reduce RTT of the latter. In fact, the neighbour set provided by Pastry as well as the possibility of parallel *GET* requests are both mandatory for the proposed cooperative replication strategy CPA, which takes into consideration interest of neighbour nodes. Also, the content-class-based replication strategy makes use of the neighbour set to place transient replicas in order to enhance query efficiency.

5.3.3.2 Pastry Implementation

The implementation of the overlay network Pastry is based on the description presented in [RD01a] and is tailored to the API of the simulation engine. The implementation covers Pastry's state tables (routing table, leaf set, neighbour set) and provides dedicated periodic simulation tasks to maintain the overlay network topology. Moreover, multiple dispatcher components dedicated to the different message types processed by the overlay network are defined. The actual core components of the implementation are (i) Pastry Node, which orchestrates the other components and interfaces with the simulation engine, and (ii) Pastry Id, which is not only used by the Pastry node but also by the simulation engine for the generation of content identifiers. The compositional structure of the overlay network including its relation to the simulation engine and the Common API provided by the latter is depicted in Figure 5.24.

The implementation of the proximity metric approximates the "distance" between any two nodes by means of the delivery time of a virtual message. This means, the delivery time of a virtual message with a fix, pre-defined size is determined by the BRITE Network given the static matrices described before. Moreover, the application interface of the Common API is extended in order to enable the distributed storage service to access both the leaf set and the neighbour set (the standard interface covers access to the leaf set, only).

Finally, the implementation of Pastry is extended to comply with the failure handling of the underlay network layer described in Section 5.3.2.4. *FAIL* messages are delivered to the distributed storage service registered with the Pastry node, to have the latter processing the failure. Notifications received in case of routing failures are directly processed by Pastry nodes. This includes re-routing of the messages to the next best node.

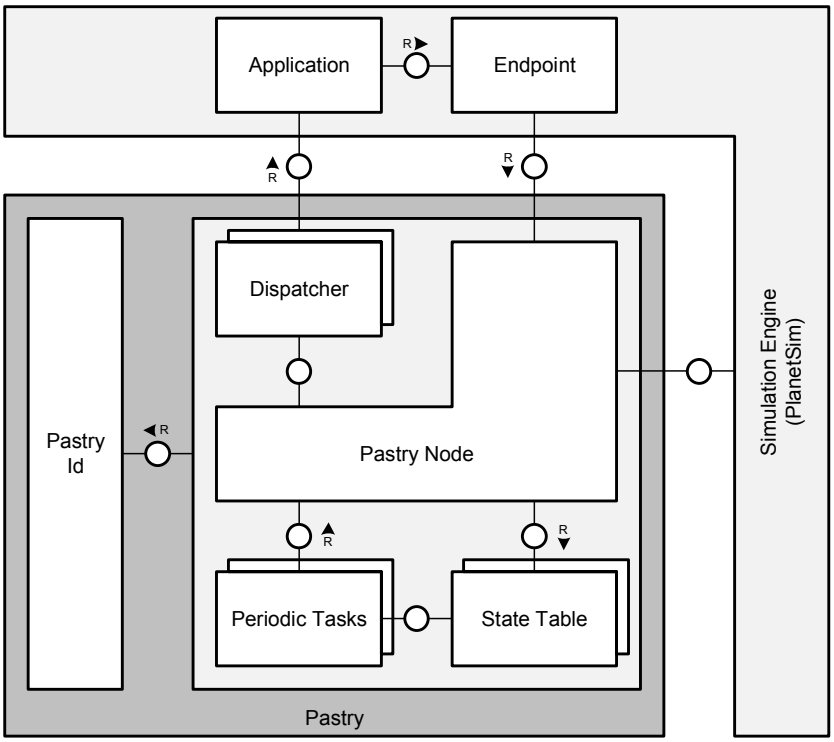


Figure 5.24: Implemented Simulation Model: Compositional Structure the of Overlay Network

Moreover, the inconsistency in the state table is addressed. This involves replacement of the failed node within the local state table as well as propagation of the failure in the overlay network to maintain the quality of the overlay network topology.

5.3.4 Distributed Storage Service

5.3.4.1 Compositional Structure

The distributed storage service provides the typical functionality of P2P storage services. Based on the dependency injection framework Google Guice[49], the distributed storage service enables plugging-in the proposed content placement strategies and can be seen as the core platform for simulating the latter. Towards the simulation engine, the service provides two main interfaces.

[49] `https://code.google.com/p/google-guice/`

The *PUT / GET / REMOVE* interface enables external access to the storage and retrieval functionality. While this interface allows other components in the simulation framework for assigning content classes to *PUT / GET* requests, the actual request processing is fully transparent to the former. For example, the requester can neither control nor infer whether the request is served locally or remotely, or whether the request triggers content replication or replacement operations.

In addition, the service includes the component Common API Handler that – as the name suggests – enables communication by implementing the application interface of the Common API as well as by dispatching messages to be processed by the distributed storage service to the corresponding component.

The compositional structure of the distributed storage service is depicted in Figure 5.25 and described in the following paragraphs. All components for which multiple strategies exist as well as the failure handling of the distributed storage service are further detailed in the following sections.

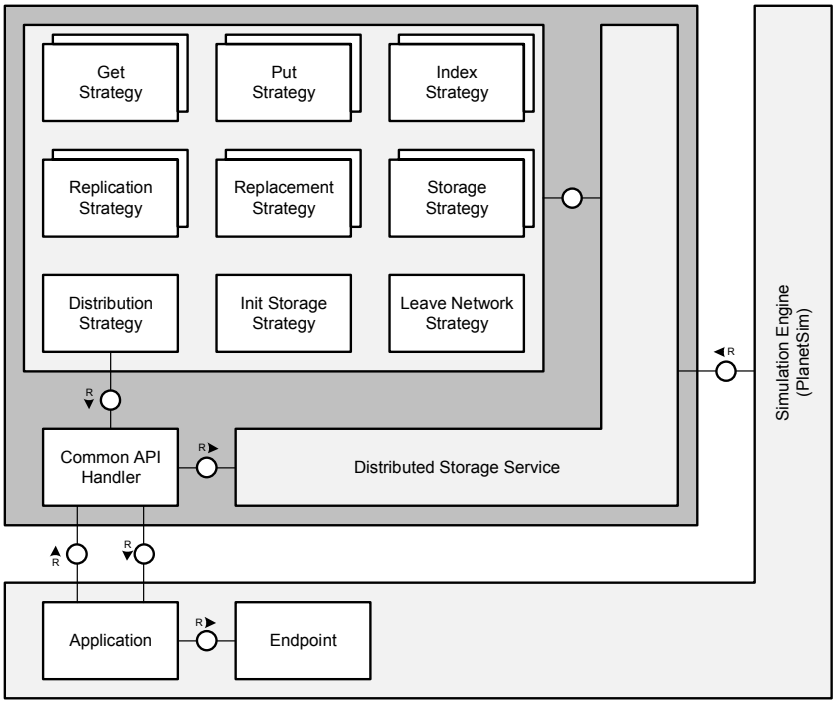


Figure 5.25: Implemented Simulation Model: Compositional Structure of the Distributed Storage Service

Get Strategy. The Get Strategy provides functionality to lookup and retrieve content stored either on-board or off-board. In addition to the content identifier, the requester has to specify the content class to enable the proposed content placement strategies to

adapt content organisation, accordingly. Content stored off-board can be retrieved by using either a pointer to the provider maintained in the index of the node or the content location and routing functionality of the overlay network implementation.

Put Strategy. The Put Strategy facilitates storage of content in the network according to the placement functionality of the overlay network implementation (see Section 5.3.3.1). In addition, the strategy enables forced on-board storage. This functionality is only applicable internally and cannot be used by external requesters. Due to the limited resources of smart products, the strategy may invoke replacement operations if content can only be stored after replacement of other objects.

Index Strategy. To enhance query efficiency and enable replication and replacement strategies (e.g., creation of local replicas after exceeding a pre-defined number of requests), the distributed storage service consists of an Index Strategy. This component is used to maintain *PUT / GET* request rates (both direct and indirect), the simulation step of the most recent content access, as well as pointers to providers if known by the local node.

Replication and Replacement Strategy. The proposed workflow-based and content-class-based replication strategies are covered by the component Replication Strategy. Similarly, the proposed replacement strategies are covered by the component Replacement Strategy. The latter is invoked by the Put Strategy if storage requests cannot be served due to other content being stored on-board. On a high level, it consists of two phases: (i) the determination of replacement candidates based on specific metrics[50] and (ii) the acknowledged replacement of these candidates to surrogate providers (i.e., next best nodes according to the placement metric applied by the overlay network implementation).

Storage Strategy. The Storage Strategy of the distributed storage service provides the interface to a virtual content store, which organises content stored on-board within a simple map.[51]

Init Storage Strategy. The actual content placement is regularly optimised by the Init Storage Strategy. It builds upon the Common API, which notifies registered overlay services of nodes joining or leaving the leaf set of the local node. In a nutshell, the strategy aims at maintaining the invariant of placing content on (surrogate) root nodes given system dynamics such as node churn. For this purpose, both nodes (i.e., the local node and the one joining its leaf set) determine content to be exchanged. The actual replacement process is acknowledged to avoid content loss. Note that neither content of class *CC_QUERY_EFFICIENCY* nor transient replicas are addressed by this strategy,

[50] LRU represents a simple metric that sorts content stored on-board in descending order of the point in time of its last access and – bottom up – chooses a set of content as replacement candidates in a way that enough storage capacity can be made available for the new content.

[51] Remember that the simulation model does only take into consideration a single persistent storage per node.

Strategy	Get Strategy	Put Strategy	Index Strategy	Storage Strategy	Replacement Strategy	Replication Strategy
1	Basic	Basic	Basic	Basic	*Basic*	MPP
2	Parallel Lookup[1]	Transient Replica[1]	Periodic[1]	Transient Replica[1]	MFR[1]	PA
3	Content Class[2]	Content Class[2]	Neighbour[2]	Remove Flag[2]	MFRTR[2]	CPA
4	MDCDN[2]	MDCDN[1]	MDCDN[1]		ECR[3]	CC
5	Caching[2]				MDCDN[1]	MDCDN

Table 5.17: Implemented Simulation Model: Strategies of the Distributed Storage Service (superscript numbers denote relationships between strategies in terms of inheritance, italicised strategies represent abstract strategies)

because they are typically stored on purpose on-board of certain nodes independent of the identifier of the nodes.

Leave Network Strategy. Likewise, the Leave Network Strategy is invoked by nodes gracefully leaving the network. It aims at replacing all content stored on-board of the leaving node to the next-best provider (content specific) to avoid content loss. Again, neither content of class *CC_QUERY_EFFICIENCY* nor transient replicas are handled by this strategy. Yet, in contrast to the Init Storage Strategy, this process is optimistic to account for the likely small time frame between announcement and the actual leave of the node under consideration.

Distribution Strategy. Finally, the Distribution Strategy decorates the Common API Handler and provides additional utility functionality. This ranges from simple extensions of overlay service messages to the determination of the destination of a message given a Pastry identifier. For example, this functionality is used by a (surrogate) root node that is not a provider of a requested content in order to identify the most suitable node maintained in its state tables to which to forward the message.

Table 5.17 presents an overview of the different strategies of the above-outlined components that can be plugged into the distributed storage service. For all components that are not covered in this table, only a single implementation exists. Note that the superscript numbers stated after most strategies indicate the related base strategy. For example, the get strategy Content Class bases upon Parallel Lookup, which itself inherits from the get strategy Basic. All strategies listed in this table are presented in the subsequent sections. These descriptions are complemented with state charts whenever beneficial. Moreover, the software structure is further detailed in Annex A.3 by means of class diagrams. Note that in most cases, the elementary functionality of the Basic strategies is already covered by the brief descriptions presented above and is not detailed any further.

5.3.4.2 Storage Strategies

In addition to the actual content, the implemented simulation model supports replicas and transient replicas. All three types include an identifier provided by the Pastry Id component of the overlay network layer as well as basic randomly assigned metadata used to generate the latter. Also, each type maintains a map with content classes assigned by different nodes and enables determination of the primary content class from the perspective of a requester (see Section 4.2.1). Both replicas and transient replicas inherit the map of content class assignments from their base object during creation. Moreover, transient replicas are explicitly assigned content class *CC_QUERY_EFFICIENCY* given their purpose of improving content access. However, there is no synchronisation of content class assignments among related content objects after content creation; each object maintains its assignment map individually.

Transient Replica. The storage strategy Basic outlined in Section 5.3.4.1 is extended by the strategy Transient Replica by means of functionality required to handle the gradually increasing TTL of transient replicas. This includes functionality to increase the TTL interval after each access as well as to transform a transient replica into a (persistent) replica after a pre-defined number of accesses was exceeded (see Section 4.1.2.3).

Remove Flag. This functionality is further extended by the strategy Remove Flag, which enables handling of lazy removal and re-initialisation properties of transient replicas. Hence, Remove Flag covers the entire life cycle of transient replicas as depicted in Figure 4.11.

5.3.4.3 Index Strategies

Periodic. The index strategy Periodic extends the index strategy Basic following the concept of temporal locality and enables weighting of *GET* requests according to the period in which requests took place. For this purpose, the strategy divides *GET* request rates maintained by the strategy Basic according to a configurable period length (see parameter t_{INDEX} explained in Section 5.4.1.2).

Neighbour. Even further, in regular intervals, the index strategy Neighbour sends the periodic index of *GET* requests extended with a snapshot of the *GET* request rates of the current period to a configurable number of nodes of the neighbour set of the local node (see parameters k and $t_{INDEX-NEIGHBOUR}$ explained in Section 5.4.1.2).

For all content stored on-board of the local node the *GET* request rate sent to its neighbours is set to a value of 0 for all periods. This is because the strategy Neighbour is solely used by and tailored to the cooperative replication strategy CPA, which incorporates the likelihood of a node receiving *GET* requests from nodes of which it is one of the best

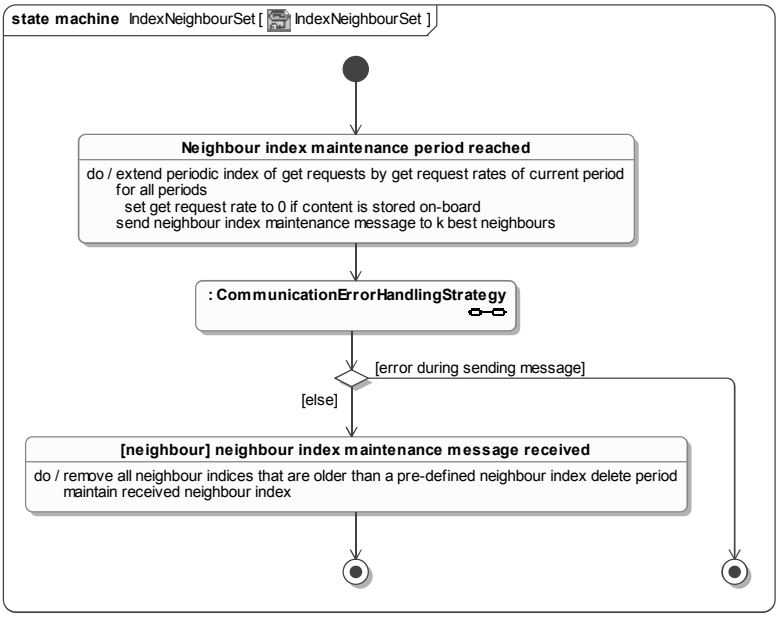


Figure 5.26: Distributed Storage Service: State Chart of the Index Strategy Neighbour

neighbours. For each of these nodes, this likelihood equals a value of 0 if content is stored on-board.

Moreover, received neighbour indices are regularly cleaned-up to avoid CPA to base decisions on out-dated information about neighbour interest. An overview of this process modelled as a state chart as well as a generic communication error handling procedure are depicted in Figures 5.26 and 5.27, respectively.

5.3.4.4 Get Strategies

Basic. The process of the get strategy Basic is illustrated in the state chart depicted in Figure 5.28. In addition to the typical characteristics of distributed content lookup and retrieval strategies it consists of the following features.

First, the strategy *aggregates direct GET requests* and provides a combined callback processing of the latter. This way, parallel handling of direct *GET* requests for the same content triggered by different local components (e.g., *GET* event and *GET* request made by the workflow engine to serve activity-related content needs) is enhanced.

Second, complementing the routing-related functionality provided by the overlay network implementation, the get strategy Basic provides means for avoiding *pointer-based message delivery loops* by maintaining intermediates of *GET* requests.

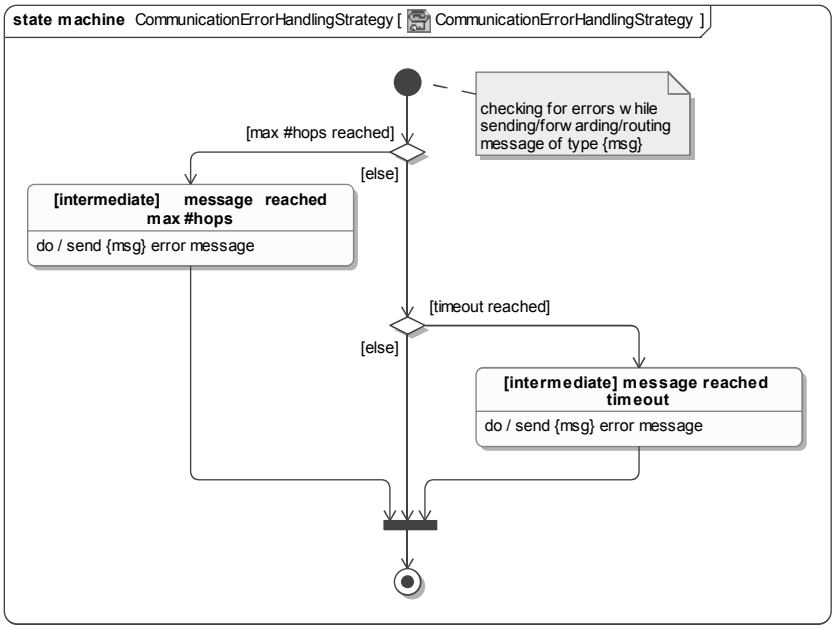


Figure 5.27: Distributed Storage Service: State Chart of the Communication Error Handling Procedure

Third, an *index invalidation mechanism* is incorporated to enhance quality of pointers maintained in the index. Each node that sends / forwards a *GET* message based on a pointer awaits an *INDEX INVALIDATION* message. This message is sent by the actual provider to the requester and all intermediates that sent / forwarded the request based on pointers maintained in their respective indices. If an *INDEX INVALIDATION* message is not received within a pre-define period, the pointer to the assumed provider is invalidated, i.e., removed from the index. If, otherwise, an index invalidation is received with a provider other than the node pointed to within the index, the latter is adapted to enhance query efficiency of subsequent *GET* requests.

Finally, the *Common API* interface is used to further enhance content lookup performance. The implementation of the *forward* method of the application interface implemented by the Common API Handler checks local availability of requested content. If the content is stored on-board, routing is aborted and the *GET* request is served by sending a *GET RESPONSE* message to the actual requester.

Parallel Lookup. The extended get strategy Parallel Lookup applies a mechanism that queries a pre-defined number of neighbour nodes of the requester in addition to the traditional request routing or sending. This approach is similar to the two-tiered lookup strategy applied by OceanStore [KWZ+00] and provides the basis for the cooperative characteristic of the proposed replication strategy CPA. If nodes were not sending *GET*

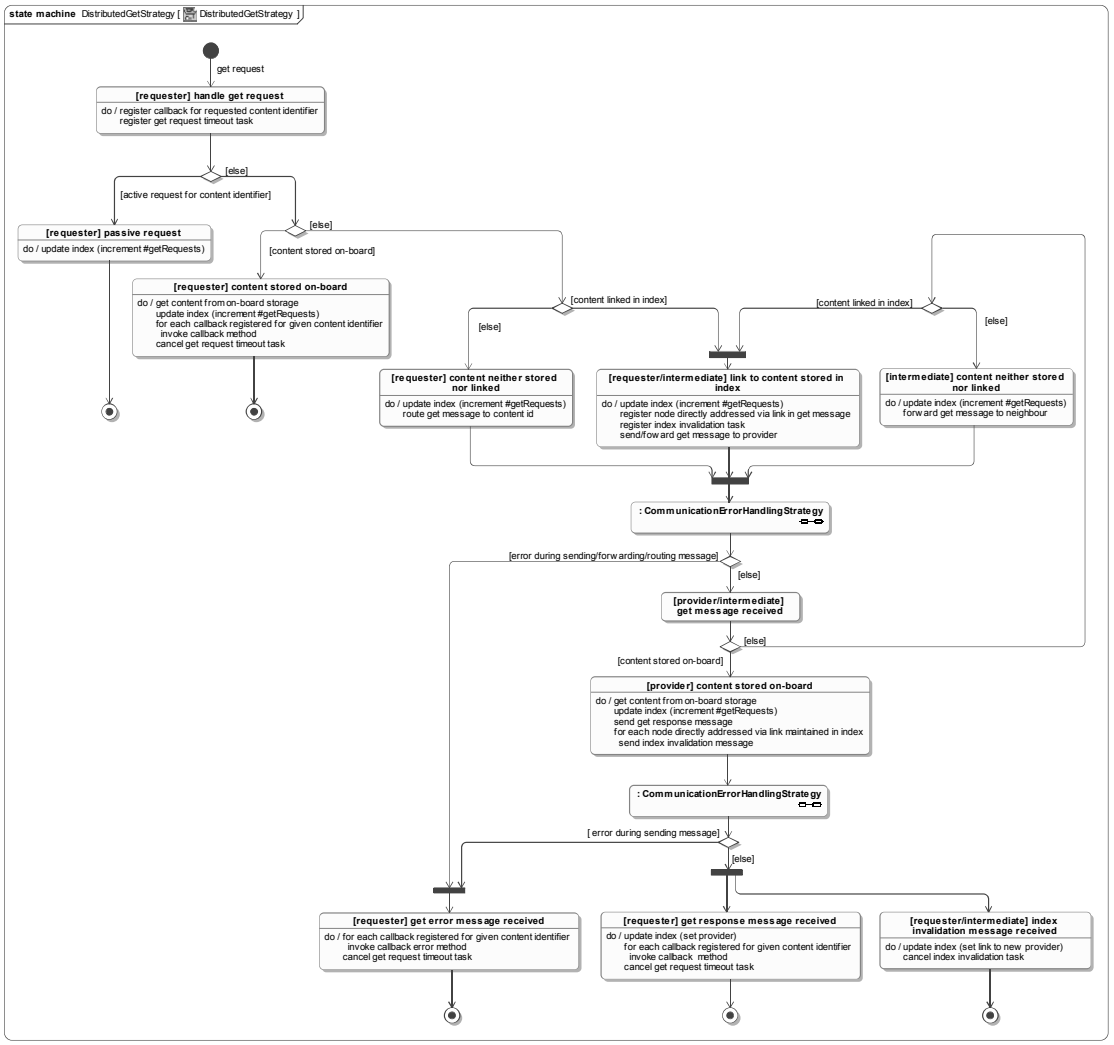


Figure 5.28: Distributed Storage Service: State Chart of the Get Strategy

requests to their neighbours, incorporating neighbour interest into the benefit metric of CPA would not add value to the accuracy of the latter.

In order to reduce the amount of additional network traffic resulting from this approach, the parallel lookup procedure is only applied to *GET* requests classified with content class *CC_QUERY_EFFICIENCY*. This adaptation of the lookup procedure subject to the content class set by the requester shows the benefit of enabling nodes to categorise their requests.

The strategy Parallel Lookup adapts the following states of the chart presented in Figure 5.28.

- [requester] handle *GET* request: In parallel to routing or pointer-based sending of a *GET* request classified as *CC_QUERY_EFFICIENCY*, the latter is sent to the k closest neighbours.
- [provider / intermediate] *GET* message received: *GET* messages received by neighbours of the requester are not forwarded but served locally by replying with either a *GET RESPONSE* or a *GET ERROR* message.
- [requester / provider] content stored on-board: If a *GET* request can be served, the content class is updated with the tuple (identifier of the requester, content class set by the requester).
- [requester] *GET ERROR* message received: In case a received *GET ERROR* message relates to a parallel lookup, it is checked whether additional replies are expected. The following states are possible.
  - This was the last message expected, invoke get error callback.
  - There are additional replies expected, wait for them.
  - The request was served already, discard the message.
- [requester] *GET RESPONSE* message received: All but the first *GET RESPONSE* message are discarded, i.e., the parallel lookup procedure is – from the perspective of the requesting component – aborted as soon as the content is retrieved.

Content Class. The third variant of the get strategy covers the specifics of the proposed replication strategy CC and incorporates the triggers associated with the latter. The strategy Content Class adapts the following states of the chart presented in Figure 5.28.

- [requester / intermediate] content neither stored nor linked & [requester / intermediate] link to content stored in index: Whenever a *GET* message is sent / forwarded by a (surrogate) root node of the requested content, the *GET* message is flagged. This flag is processed by the actual provider and enables maintenance of replica placement in the leaf set of the provider as described in Section 4.2.2.

- [requester / provider] content stored on-board: If (i) the provider is (surrogate) root node for the requested content or (ii) the *GET* message is flagged as described above, then the mechanism of the replication strategy CC for maintaining replica placement in the leaf set of the provider is triggered.
- [requester] get response message received: If the number of requests for the given content exceeds a pre-defined threshold, a local replica is created. Moreover, if the primary content class related to the requester equals *CC_QUERY_EFFICIENCY*, then the mechanism of the replication strategy CC for replica placement in the neighbour set of the requester is triggered.

5.3.4.5 Put Strategies

Basic. The process of the put strategy Basic is illustrated in the state chart depicted in Figure 5.29.[52] As outline in Section 5.3.4.1, the strategy Basic supports content storage in the network according to the content placement approach of the overlay network implementation and provides means for forcing on-board content storage. The second functionality is dedicated to internal use and cannot be utilised via the API of the distributed storage service. As an example, forced on-board content storage is used for pushing replicas as well as for replacing content triggered by the leave network strategy. Moreover, the put strategy Basic makes use of the replacement strategies presented in the next section to enable serving content storage requests after "outsourcing" other content objects. In addition to the typical characteristics of distributed put strategies the strategy Basic consists of the following features.

First, if content is to be stored on-board of a node that is already provider of this content, the strategy applies a *content quality check*. That is, the strategy checks whether the content object stored on-board is of "less quality" than the content object requested to be stored on-board. If so, the content stored on-board is overwritten with the new content object. For example, if a transient replica is stored on-board and a content or replica with the same identifier as the transient replica is requested to be stored, than the transient replica is overwritten by the new object. The general transitive quality order is *content overwrites replica overwrites transient replica*. Note that forced on-board storage is served in either case (i.e., *PUT RESPONSE* message is sent). However, if content is requested to be stored on a node other than the requester and content was not overwritten (i.e., content stored on-board of destination is of "better quality" than content requested to be stored), then the request is forwarded to the next best node.

Second, the mechanisms that check (i) if content can be stored on-board and (ii) if content can be stored after replacement both verify whether the given content is handled

[52] For reasons of simplicity, the description of the put strategy uses the term "requester" to specify the node / role triggering the operation, i.e., requesting to store a content object.

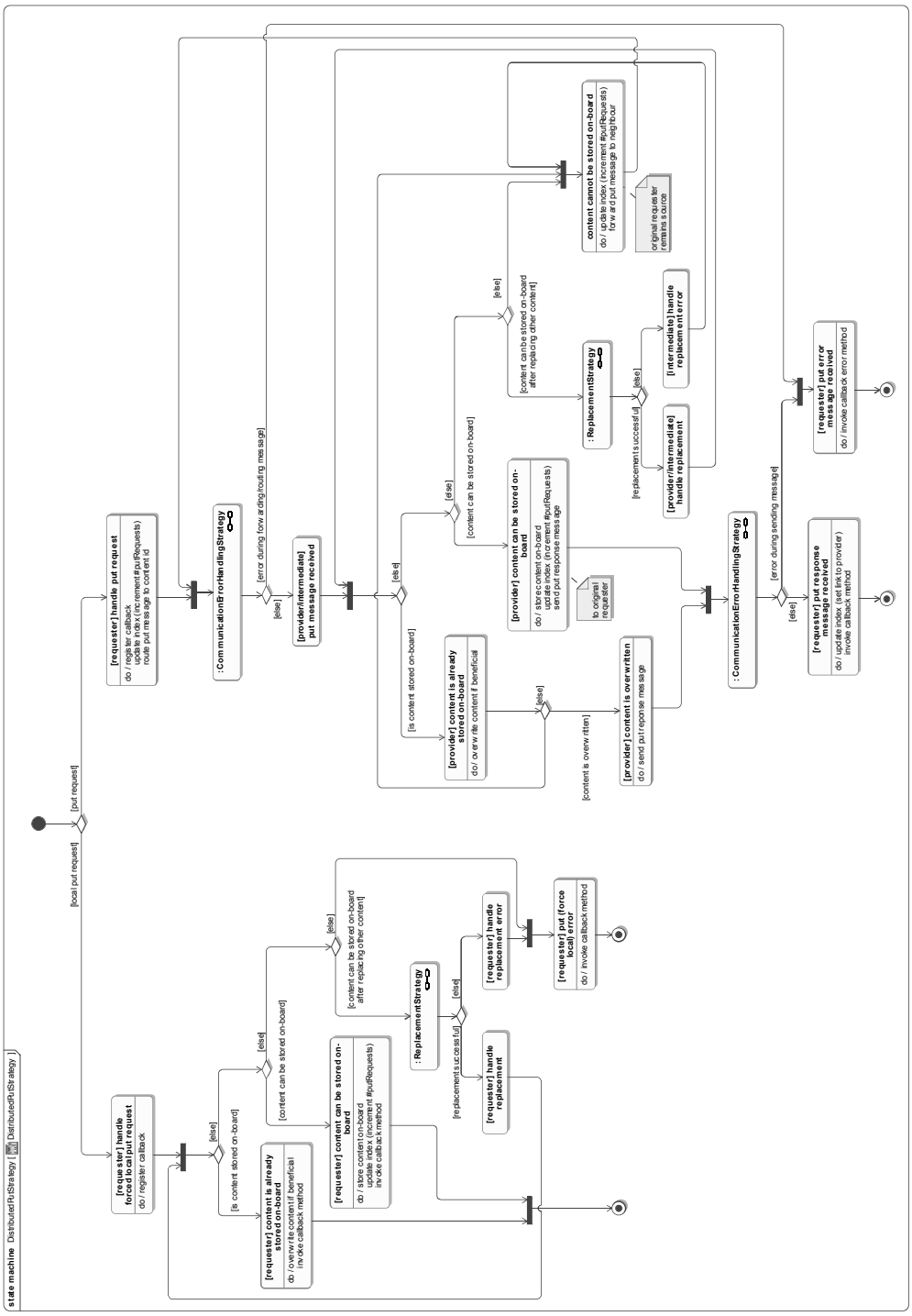


Figure 5.29: Distributed Storage Service: State Chart of the Put Strategy

by an *active replacement process*. If so, the *PUT* request cannot be served locally and is forwarded to the next best node.

Transient Replica. The strategy Transient Replica extends a single state of the chart depicted in Figure 5.29 in order to enable handling of transient replicas.

- [requester] content can be stored on-board: If a transient replica is stored on-board, then the simulation task associated with controlling the gradually increasing TTL of the latter is initialised.

Content Class. This variant of the put strategy includes all extensions required to trigger placement and maintenance of the proposed replication strategy CC. It extends the following states of the chart depicted in Figure 5.29.

- [requester] handle put request: This extension covers direct *PUT* requests and enables triggering generation and placement of transient replicas if the primary content class related to the requester equals *CC_QUERY_EFFICIENCY*. This includes (i) placement of a transient replica on-board of the requester if the latter is not already provider for this content and if there is no active replacement process for the latter (see above) as well as (ii) placement of transient replicas in the neighbour set of the requester.
- [provider] content can be stored on-board: This method is invoked after receiving an indirect *PUT* request as well as after successfully replacing other content objects in order to serve an indirect *PUT* request. The node on which content is initially stored (i.e., after being "injected" into the network) is referred to as primary provider (see Section 2.3.1). Hence, according to the configuration of the proposed replication strategy CC, a certain number of replicas is placed in the leaf set of the primary provider depending on the primary content class related to the latter.

5.3.4.6 Replacement Strategies

Basic. The process of the implemented (abstract) replacement strategy Basic is depicted in Figure 5.30. As stated afore, the overall replacement concept aims at making available storage capacity to serve a given *PUT* request by replacing, i.e., "outsourcing", content stored on-board to the next best node given the identifier of this content.[53] The strategy monitors the replacement process for each replacement candidate and does not remove the instance stored on-board until an *acknowledgement* of the successful storage on-board of the new provider is received in terms of a *REPLACE RESPONSE* message. If not enough storage capacity can be made available – indicated by the node triggering the replacement process receiving at least one *REPLACE ERROR* message –, then the strategy

[53] Of course, the question whether the new content can be stored on the target node at all (i.e., content size $\leq$ node storage capacity) is checked in a first step.

supports a pre-defined number of *iterations* (set to a value of 2 by default), which reruns the entire replacement process.

The replacement process is *non-recursive*, i.e., the depth of the replacement process is limited to a single node. While the receiver of a *REPLACEMENT* message can forward the latter to the next best node within its leaf set in case it is not able to store the replacement candidate on-board, it cannot trigger another replacement process to make available the required storage capacity. This design improves replacement efficiency by avoiding complex replacement operations for serving a single *PUT* request given the timeout properties of the latter.

Despite the acknowledged approach, the overall process is *optimistic* and there is *no rollback* functionality. For example, if a number of 3 content objects are to be replaced and a single one fails, then the successfully replaced content objects remain placed on-board of the new provider. While this might result in a worse placement, it reduces complexity of the replacement process. Moreover, the acknowledgement enables the node triggering the replacement process to maintain a pointer to the new provider in its local index thereby limiting the impact on query efficiency of future *GET* requests.

The actual *strategies for determining replacement candidates* can be plugged-in (see state "[requester] determine content to be replaced" in Figure 5.30) and are implemented according to the strategies presented in Section 4.3. All strategies verify two aspects prior to classifying content as replacement candidate. First, similar to the proposed put strategy implementation, content can only be replaced if it is not handled by another replacement or storage initialisation process. Second, to avoid high levels of dynamics, content cannot be replaced but after a pre-defined initial delay after it was stored on-board of its current provider (set to a value of 50 simulation steps by default).

MFR. The replacement strategy MFR adopts the concept proposed by [KRT06, KRT07] and makes use of the index to determine replacement candidates. Content stored on-board is sorted in ascending order of the ratio between access rate and content size with content objects with identical quotients being sorted in ascending order of the simulation step when they were last accessed. To determine replacement candidates, the ordered queue is processed head to tail until enough storage capacity can be made available for serving the pending *PUT* request.

MFRTR. The implementation of MFRTR extends MFR by taking into account transient replicas flagged to be removed. Hence, as described in Section 4.3.1, MFRTR tries to make the required storage capacity available by removing transient replicas flagged to be removed prior to replacing content. Note that the actual removal of transient replicas flagged to be removed is only performed if (i) storage capacity can be made available by removing transient replicas flagged to be removed, only, or (ii) if the remaining required storage capacity can be made available by replacing other content objects.

ECR. Finally, the strategy ECR extends the replacement candidate determination functionality of MFRTR by taking into consideration content classes. Determination of con-

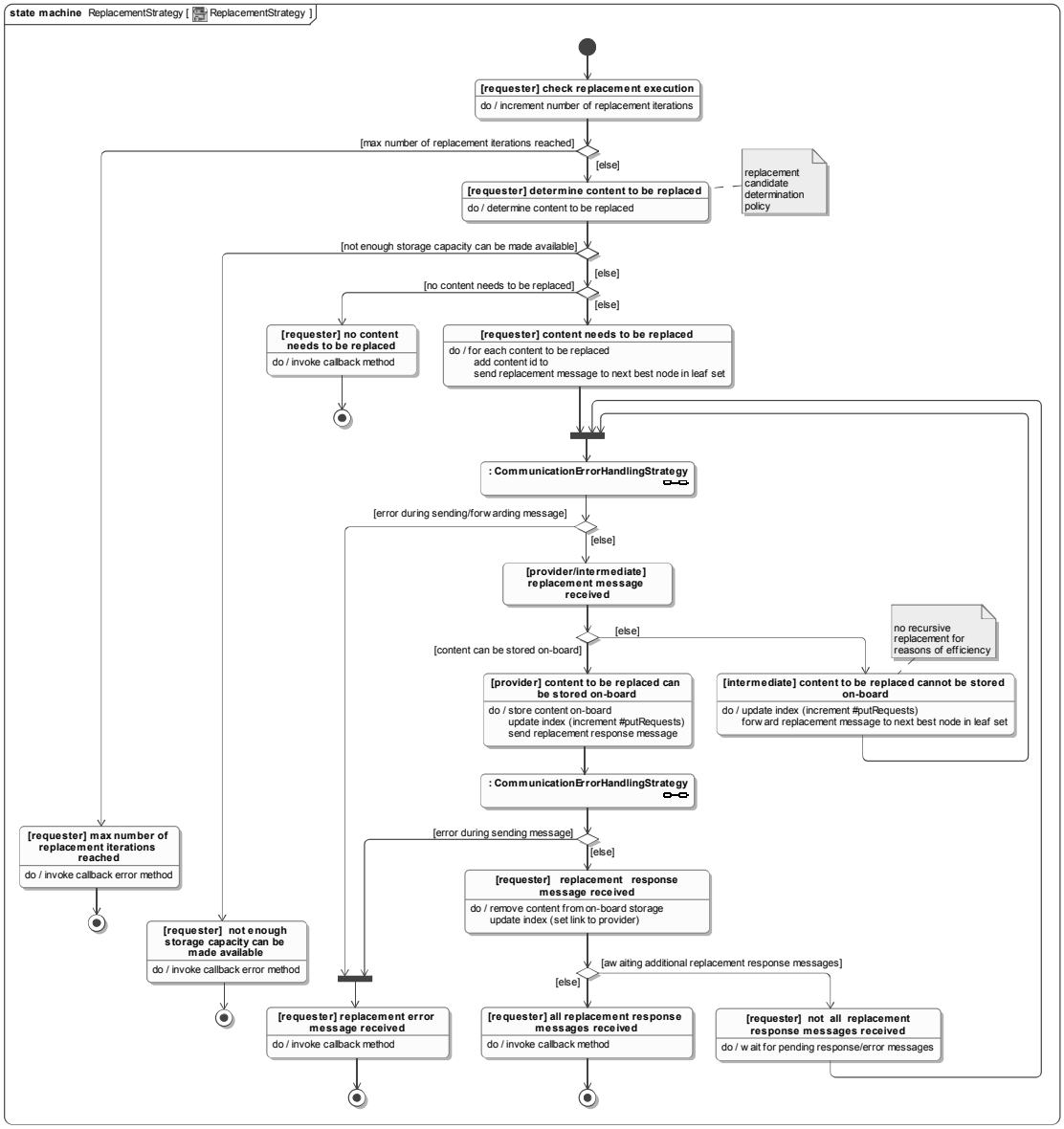


Figure 5.30: Distributed Storage Service: State Chart of the Replacement Strategy

tent to be replaced includes case-by-case strategies depending on the primary class of the content to be stored on-board as well as of the potential replacement candidates (both from the perspective of the node processing the replacement operation). While candidates of primary content class with "less quality" (see Section 4.3.2) than the new content can directly be taken as replacement candidates, the strategy makes use of the index strategy Periodic to calculate the cost / benefit ratio presented in Section 4.3.2 in case of class equality.

5.3.4.7 Replication Strategies

The component Replication Strategy of the distributed storage service covers the different variants of the proposed workflow- and content-class-based replication strategies. The component itself consists of multiple services that can be plugged-in depending on the strategy to be simulated. An overview of the component as well as its services is presented in Figure 5.31.

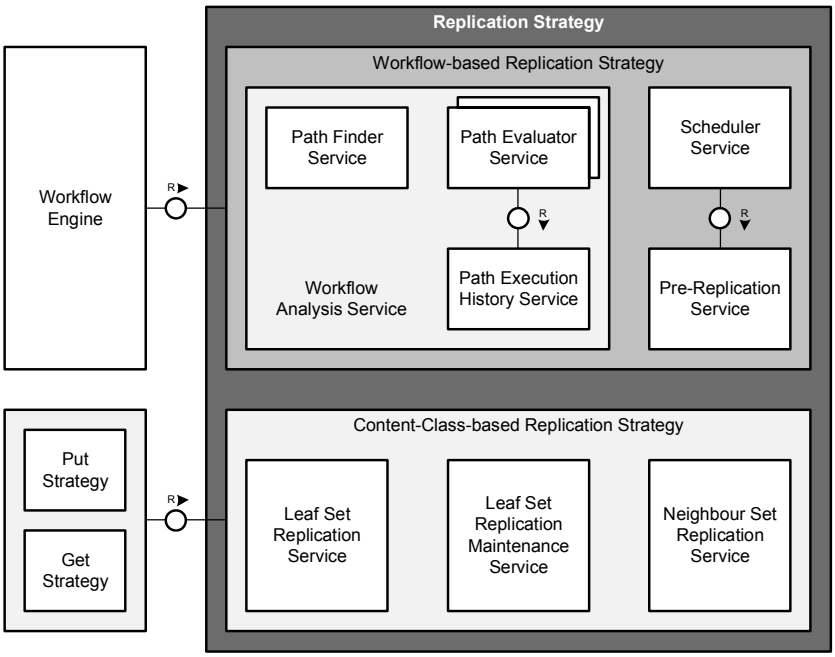


Figure 5.31: Implemented Simulation Model: Compositional Structure of the Replication Strategy

Workflow-based Replication Strategy

The Workflow-based Replication Strategy consists of and orchestrates services that cover the three phases of the proposed workflow-based replication strategies MPP, PA, and CPA.

Workflow Analysis Service. The workflow analysis phase is implemented by the Workflow Analysis Service, which encapsulates the following three services.

- The Path Finder Service enables determination of all paths of a given workflow starting from an activity specified by the caller and implements the restrictions described in Section 4.1.2.1 (i.e., reduction of path length to a maximum number of α activities following the first XOR transition, reduction of path length to only include a single XOR transition).
- In order to enable taking into account temporal locality of workflow processing, activity execution is tracked by the Path Execution History Service. This service provides functionality to determine periodically-weighted path execution probabilities as well as to identify the path with the highest weighted execution probability (i.e., the most probable path).
- The Path Evaluator Service encapsulates the path assessment metrics $f^{MPP}(p,T)$, $f^{PA}(p,T)$, and $f^{CPA}(p,T)$ of the three proposed workflow-based replication strategies. While MPP solely relies on the path with the highest execution probability, PA and CPA incorporate the index strategies Periodic and Neighbour, respectively, to enable their extended path assessment approach. As a result, the Path Evaluator Service provides a set of paths to be further processed by the strategy sorted in descending order of their assessment values.

Scheduler Service. The Scheduler Service implements the scheduling strategy presented in Section 4.1.3.2. This comprises scheduling of content pre-replication requests as well as schedule adaptation functionality to realise linkage between schedule and progress in the control flow of the workflow for which content is being pre-replicated.[54]

Pre-Replication Service. The Pre-Replication Service processes the schedule provided by the Scheduler Service to enable retrieval and local placement of activity-related content needs. Thus, the service tries to created a transient replica (content class *CC_QUERY_EFFICIENCY*) for each content listed within the schedule that is not stored on-board.

Strategy Orchestration. Finally, the Workflow-based Replication Strategy includes an orchestration component. This component is invoked by the workflow engine prior to activity processing and facilitates pre-replication scheduling subject to the progress in the

[54] A single implementation of this service is reasonable for all strategies. The extended scheduling functionality of PA and CPA is not applied in MPP which solely takes into account a single path to schedule pre-replication requests of activity-related content needs for.

control flow of the workflow. The orchestration functionality consists of the following steps. First, the path execution history is maintained (on activity level) by the Path Execution History Service. Second, it is checked whether (an iteration of) the replication strategy is to be executed (i.e., start activity of a workflow or an activity following an XOR transition).

If so, any active pre-replication related to the given workflow is stopped and the related schedule is cleared. Thereafter, available paths are determined (Path Finder Service) and assessed (Path Evaluator Service). For all paths identified for further handling, the pre-replication schedule is generated (Scheduler Service) and processing of the latter is started (Pre-Replication Service).

Else, if the replication strategy is not to be executed, the schedule is adjusted and it is checked whether re-scheduling should be done depending on the progress in the control flow. If so, the schedule is extended, accordingly.

Content-Class-based Replication Strategy

The Content-Class-based Replication Strategy consists of three services that cover the functionality described in Section 4.2.2. This section focuses on the implementation of the services, only. The invocation of the latter done by the Content Class variants of the put and get strategies is described in the related sections and is not further detailed.

Leaf Set Replication Service. The initial content replication in the leaf set of the primary provider is covered by the Leaf Set Replication Service (see state "[provider] content can be stored on-board" of the put strategy Content Class presented in Section 5.3.4.5). This service places replicas on its successors based on the assumption that it is the most suitable node given the content location properties of the overlay network implementation (push-based by means of a *REPLICATE* message consisting of a locally created replica). The successors (i.e., receivers of the *REPLICATE* message) make use of the put strategy to force on-board storage of the new replica. The actual number of successors on which to place replicas depends on the primary content class of the content from the perspective of the primary provider.

Leaf Set Replication Maintenance Service. The Leaf Set Replication Maintenance Service, maintains number and placement of replicas in the leaf set of the node processing the maintenance operation (see state "[requester / provider] content stored on-board" of the get strategy Content Class presented in Section 5.3.4.4). For this purpose, the latter sends *REPLICATE PING* messages to a pre-defined number of nodes in its leaf set (i.e., potential providers of replicas) with the number of nodes being addressed depending on the primary content class from the perspective of the sender. Each receiver replies with a *REPLICATE PING RESPONSE* message indicating whether it is a provider of the given content. Using a simulation timer task, the node processing the maintenance operation handles each *REPLICATE PING RESPONSE* message received. If the number of providers exceed a threshold, then the maintenance operation is aborted. Else, if

(i) all expected *REPLICATE PING RESPONSE* messages are received and the number of providers is below the threshold, or (ii) the waiting period is passed, then the node starts placement of new replicas (push-based using *REPLICATE* messages). This way, replicas are distributed until the minimum number of replicas given the primary content class from the perspective of the node processing the maintenance operation is met again.

Neighbour Set Replication Service. The Neighbour Set Replication Service takes care of placing transient replicas for content of primary content class *CC_QUERY_EFFICIENCY*. Hence, transient replicas are placed on-board and in the neighbour set of initiators of *PUT* requests (see state "[requester] handle put request" of the put strategy Content Class presented in Section 5.3.4.5) as well as initiators of *GET* requests in case of the request rate exceeding a pre-defined threshold (see state "[requester] get response message received" of the get strategy Content Class 5.3.4.4). Again, (in this case, transient) replicas are pushed to nodes in the neighbour set by means of *REPLICATE* messages.

5.3.4.8 Messages and Failure Handling

As described in Section 5.3.3.2, the distributed storage service provides means for handling *FAIL* messages that are sent by the underlay network implementation in case a recipient of a sent or forwarded message left / failed prior to message delivery. *FAIL* messages represent container messages, which encapsulate the actual message that failed to being delivered.

Table 5.18 lists the message types used by the distributed storage service as well as failure states and failure handling applied by the latter. Note that there is no failure handling in place for the case in which the source of a message failed prior to message delivery – such messages are simply dropped by the failure handling of the underlay network implementation (see Section 5.3.2.4).

Each of the listed messages is assigned a maximum number of overlay hops as well as a timeout value in terms of simulation steps (see Section 5.4.1 for configuration details). This is required because a single overlay hop may last multiple simulation steps given the delay in the underlay network implementation. Both values are evaluated in the forward method of the Common API Handler of the distributed storage service. In case any of the attributes exceeds the message-assigned threshold values, the message-related operation is aborted and – in case of *PUT*, *GET*, or *REPLACE* operations – a request-related error message is returned to the source of the request.

5.3.5 Workflow Engine

The implemented simulation model consists of a workflow engine that enables operation of the proposed workflow-based replication strategies. The engine is tailored to the

Message	Description	Failure	Failure Handling
FAIL	sent by underlay network implementation in case a recipient of a sent or forwarded message left / failed prior to message delivery	send, destination unavailble	Drop message
GET	get strategy, indirect get request	send, destination unavailble	update index and route GET message
		forward, destination (in node's leaf set) unavailable	forward GET message to next best neighbour, send GET ERROR message to source if no neighbour can be determined
		forward, destination (indicated by pointer in index) unavailable	update index and route GET message
GET ERROR	get strategy, error serving indirect get requests	send, destination unavailble	Drop message
GET RESPONSE	get strategy, successfully served indirect get requests including requested content	send, destination unavailble	Drop message
INDEX INVALIDATION	get strategy, index invalidation including actual provider	send, destination unavailble	Drop message
INIT STORAGE	init storage strategy, init storage requests with available storage capacity	send, destination unavailble	Drop message
INIT STORAGE RESPONSE	init storage strategy, set of content to be replaced	send, destination unavailble	Drop message
INIT STORAGE ACK	init storage strategy, acknowledgement of set of content that was replaced	send, destination unavailble	Drop message
LEAVE NETWORK	leave network strategy, set of content to be replaced	send, destination unavailble	Drop message
NEIGHBOUR INDEX	neighbour index strategy, periodic index of source	send, destination unavailble	Drop message
PUT	put strategy, indirect put request including content to be stored	forward, destination unavailable	forward PUT message to next best neighbour, send PUT ERROR message to source if no neighbour can be determined
PUT ERROR	put strategy, error serving indirect put request	send, destination unavailble	Drop message
PUT RESPONSE	put strategy, successfully served indirect put requests	send, destination unavailble	Drop message
REPLACE	replacement strategy, content to be replaced	send, destination unavailble	forward REPLACE message to next best neighbour, send REPLACE ERROR message to source if no neighbour can be determined
		forward, destination unavailable	
REPLACE ERROR	replacement strategy, error serving indirect get requests	send, destination unavailble	Drop message
REPLACE RESPONSE	replacement strategy, successfully served indirect get requests	send, destination unavailble	Drop message
REPLICATE	replication strategy, push (transient) replica to destination	send, destination unavailble	Drop message
REPLICATE PING	content-class-based replication strategy, check whether destination is provider of (transient) replica	send, destination unavailble	Drop message
REPLICATE PING RESPONSE	content-class-based replication strategy, indication whether source is provider of (transient) replica	send, destination unavailble	Drop message

Table 5.18: Failure Handling of the Distributed Storage Service

purpose of the evaluation study. Hence, workflow execution is fully automated and there is no user interaction or application logic included in the control flow. Activity execution covers access to activity-related content needs, which is enabled by the get strategy of the distributed storage service, only. Moreover, activities are assigned a minimum and a maximum processing time; the actual processing time within these boundaries depends on the time required to retrieve activity-related content needs. As described in Section 5.3.4.7, the workflow engine is interlinked with the workflow-based replication strategy to enable pre-replication of activity-related content needs.

The workflow engine supports a simplified workflow modelling approach. This approach enables definition of activity-related content needs as well as workflow structures with XOR transitions. All outgoing transitions of an XOR transition are assigned transition probabilities that follow a normalised Gaussian distribution. The dynamic selection of one of the alternative paths to be followed is realised by means of a finite range discrete distribution. According to the discussion presented in Section 4.1.1, AND and OR transitions are not supported.[55]

Each node in the simulation model that is capable of processing workflows (see Section 5.2.5 for an exemplary scenario) is assigned an instance of the workflow engine that orchestrates the workflow execution environment. Each of these workflow engines consists of a workflow factory that enables generation of a pre-defined number of workflows (see Section 5.4.1) randomly chosen from the pool of workflows associated with the evaluation scenario (see Figure 5.17 as an example). Similar to the modelling of XOR transitions described above, these workflows are assigned execution probabilities that follow a normalised Gaussian distribution; the selection of a workflow to execute follows a finite range discrete distribution.

Hence, all nodes able to execute workflows make use of random subsets of the overall pool of workflows. Yet, the transition probabilities assigned to XOR transitions are dynamically configured per node. Similarly, the actual content set needed per activity is chosen randomly out of the content stored in the simulated network (note that the number of content objects remains unchanged). Even further, to take into consideration node churn and related system dynamics, content needs are verified and adjusted prior to workflow execution to ensure that they can be served theoretically.

The actual retrieval of activity-related content needs during activity execution is managed by a simulation task. This task limits retrieval time of activity-related content needs according to the maximum activity execution time mentioned above. However, in order to avoid blocking / cancelling workflow execution, the latter is continued even in case not all of the activity-related content needs could be served during the defined time period. Given the purpose of the evaluation study as well as the quality attributes defined in Section 5.2.4, this approach is feasible. The evaluation is not performed on the num-

[55] An enhanced workflow modelling approach was developed in the course of the EU-funded research project SmartProducts and is presented in [MSB11].

ber of completed activities or workflows, but on individual content objects / requests as well as the distribution of corresponding results over simulation time (e.g., RTT and hit rate of *GET* requests).

5.3.6 Implementation of Related Work

To broaden the evaluation of the proposed content placement strategies, three additional replication strategies were realised as part of the implemented simulation model. According to the results of the study on state-of-the-art content placement strategies presented in Section 3.2, MDCDN is used as representative for active replication strategies that apply access pattern recognition for estimating upcoming demand. In addition, a basic caching-like strategy that assumes temporal locality and purely relies on past observations is utilised as representative for reactive replication strategies. Finally, a strategy without replication functionality is configured to provide a baseline for the evaluation study.[56]

5.3.6.1 MDCDN

In order to facilitate simulation and evaluation of MDCDN, the latter was transformed into a P2P overlay service. All adaptations are based on the pseudo code modelling of MDCDN presented in [AMAM04]; an overview of the parameters used in the following description is shown in Algorithm 5.4.

Adaptation of MDCDN

Notes on Effort Calculation. The P2P version of MDCDN applies partial replication. For this reason, the amount of bytes transferred during replication equals the amount of bytes transferred during indirect service for a certain content object. This leads to the relation between the effort for serving predicted demand for content not stored on-board (bi^c) and the effort for retrieving a replica of the latter (br^c) shown in Algorithm 5.5.

Moreover, because of the simulation model assuming immutable content, there is no effort for maintaining consistency, i.e., $bm^c = 0$ for all content objects. However, this would lead to nodes creating replicas for which they are not provider if $d^{ic} > 1$. Similarly, replicas would only be removed if there would be no predicted demand in the upcoming δ periods, i.e., $bi^c \leq 0$.

In order to optimise replication / removal boundaries and avoid high replication degrees, bm^c has been realised as a configurable value (see Section 5.4.1.3). This results

[56] Note that the novel replacement strategies are assessed in comparison with MFR – their base strategy – only.

Algorithm 5.4: Parameters used in the Active Replication Strategy MDCDN

1: i,j := node
2: c := content
3: δ := #forecast periods
4: d^{ic} := predicted demand for c at i in the upcoming δ periods
5: d^{ic}_j := predicted indirect demand of node j at node i in the upcoming δ periods
6: sr^c := #bytes transferred during replication of c
7: si^c := #bytes transferred during indirect service of c
8: bi^c := traffic generated for serving predicted demand if c not stored locally
9: br^c := traffic generated for retrieving a replica of c
10: bm^c := traffic generated for maintaining a replica of c
11: bm^c_i := traffic generated for node i maintaining a replica of c
12: m^{tc} := indicates whether c is modified during period t
13: $load_i$:= overall predicted indirect demand at node i in the upcoming δ periods
14: T_l := threshold to control indirect demand replication

Algorithm 5.5: Effort Relations in Active Replication Strategy MDCDN

1: $sr^c = si^c$
2: $bi^c = d^{ic} \times br^c$
3: $bi^c > br^c$, if $d^{ic} > 1$

in replicas being removed earlier as well as higher predicted demand being required for the creation of new replicas (see [AMAM04]).

Incorporation of Limited Storage Capacity. In order to account for the limited storage capacity of nodes in the simulation model, predicted demand d^{ic} is sorted in descending order to use the limited storage capacity for replicas with the highest predicted demand. Even further, the replica removal procedure is processed prior to the replication procedure to avoid unnecessary content replacement.

The replication of content from the "closest server" is fully covered by the underlying P2P overlay network, i.e., the replication request (a *GET* message followed by storing a replica on-board) is routed to the provider or sent if a pointer is maintained in the local index.

Sequential Phase Processing. While the two phases (replication based on predicted direct demand and replication based on predicted indirect demand) are processed iteratively per content in the original MDCDN algorithm, they are handled sequentially in the P2P version. This is because content changes caused by the replication strategy are not effective at the simulation step to which the strategy is executed. Consequently, the prediction of indirect demand is identical in all iterations.

Handling of Predicted Indirect Demand. Since the while loop approaching indirect demand seems to be incomplete in [AMAM04] (e.g., load is never decremented), it has been adapted according to the pseudo code shown in Algorithm 5.6. Moreover, it is assumed that there is a typo in the pseudo code in that they mean $if(bi^c > bm^{cj})\ then$ instead of $if(d_j^{ic} > bm^{cj})\ then$. Otherwise, the calculation of bi^c would not be meaningful. Moreover, due to the ordering of d_j^{ic} and the decision of not considering content updates in the simulation model, the adaptation of the while loop does neither consider the calculation of bi^c and bm^{cj} nor the related if statement. Finally, the load of the node is decremented by d_j^{ic} when content c is replicated to node j. As soon as the load is below the threshold T_l, the procedure for handling indirect demand is aborted.

Algorithm 5.6: Adaptation of Active Replication Strategy MDCDN

```
1: if load_i < T_l then
2:     return                                   ▷ no need for replication
3: end if
4: Sort d_j^ic in descending order              ▷ highest predicted indirect demand first
5: for all d_j^ic do
6:     if content c has not been replicated then
7:         replicate content c to node j        ▷ node with highest indirect request
8:         load_i = load_i − d_j^ic
9:         if load_i < T_l then
10:            return
11:        end if
12:    end if
13: end for
```

Implementation of MDCDN

The implementation of MDCDN uses specific strategies of the following components of the distributed storage service.

Index Strategy MDCDN. In order to enable separate prediction and handling of direct and indirect demand, MDCDN incorporates the index strategy MDCDN. This index strategy differentiates direct and indirect *GET* requests, which are both indexed periodically as described in [AMAM04] with configurable period length (see Section 5.4.1.3).

Get Strategy MDCDN. To properly maintain this index, there is a dedicated get strategy MDCDN, which extends the strategy Parallel Lookup in order to ensure comparability with the proposed replication strategies in terms of query efficiency. This get strategy extends the two states "[requester] handle get request" and "[provider / intermediate] get message received" (see Section 5.3.4.4) to maintain both direct and indirect request rates, respectively.

Put Strategy MDCDN. Similarly, the put strategy MDCDN includes minor adaptations of the states "[requester] put response message received" and "[requester] put error message received" (see Section 5.3.4.5) to enable proper index maintenance.

Replacement Strategy MDCDN. The replacement strategy MDCDN uses the predicted direct demand as primary means for determining replacement candidates. Thus, content is sorted in ascending order of predicted direct demand. If content objects show identical demand prediction, then the overall request rate and the simulation step of the last access are used as comparison criteria 2 and 3, respectively. This active replacement strategy ensures that content with high expected direct demand is kept with the node as long as possible.

Replication Strategy MDCDN. The core functionality of MDCDN is implemented as part of its replication strategy. This strategy covers the algorithm presented in [AMAM04] as well as the above-stated adaptations. Amongst others, this includes prediction of direct and indirect demand using the statistical demand forecasting method Double Exponential Smoothing with configurable smoothing factor and forecast period values. This prediction makes use of the adapted index strategy (see above) and is used as basis for optimising content placement.

5.3.6.2 Caching

The strategy Caching represents a basic reactive replication strategy that assumes temporal locality and purely relies on past observations for making replication decisions. It uses the index strategy of the distributed storage service to maintain *GET* request rates. If this rate exceeds a configurable threshold for a certain content, then a replica of the latter is created and stored on-board in order to enhance query efficiency of future requests.

From an implementation point of view, the strategy Caching makes use of a dedicated get strategy, which adapts the state "[requester] get response message received" (see Section 5.3.4.4), only. In this state, the *GET* request rate of the requested content is compared with the defined threshold. If the threshold is exceeded, a new local replica is created (if the content is not stored on-board, according to the aforementioned state). Again, the get strategy Caching inherits the strategy Parallel Lookup to ensure comparability with the proposed replication strategies in terms of query efficiency.

5.3.6.3 No Replication

The strategy No Replication does not provide or use any means for content replication. It fully relies on the basic strategies of the distributed storage service and is used as baseline for the evaluation of the proposed replication strategies to analyse absolute enhancements given the quality attributes presented in Section 5.2.4.

5.4 Simulation Data and Evaluation Results

This section presents the setup of the simulation model given the evaluation scenario defined in Section 5.2.5. This includes different configurations of the distributed storage service (i.e., the assembly of the strategies described in Section 5.3.4), the parameter setting of the proposed content placement strategies and related work, as well as the configuration of the workflow engine. Moreover, the setup of the messages used by the distributed storage service as well as the configuration of the overlay network implementation Pastry are presented. Finally, the simulation results are analysed in detail based on the quality attributes defined in Section 5.2.4 in order to reveal the actual performance of the proposed content placement strategies.

5.4.1 Setup and Configuration

5.4.1.1 Strategy Assembly

An overview of the configurations of the distributed storage service is presented in Table 5.19. This table covers 10 different combinations of the proposed replication and replacement strategies named DSF4SP_1 to DSF4SP_10 as well as the strategy assembly of related work described in Section 5.3.6. In addition to the novel strategies, the table presents the get, put, index, and storage strategies plugged in the distributed storage service for each strategy assembly.

For example, DSF4SP_10 combines the replication strategies CPA and CC, and applies the replacement strategy ECR. As described in Sections 5.3.4.4 and 5.3.4.5, the Content Class strategies of the get strategy and the put strategy are needed to support the content-class-based strategies CC and ECR. Moreover, ECR implies the storage strategy Remove Flag to cover the entire replica life cycle (see Figure 4.11). Finally, the index strategy Neighbour is required to support the cooperative nature of CPA, which takes into account neighbour interest to improve accuracy of future demand prediction.

Note that out of the overall number of 24 possible combinations (note that (i) the proposed replication strategies can be evaluated individually and (ii) MFR is used as base replacement strategy), only a subset of 10 combinations are simulated. This is for the following reasons. First, configurations without any of the proposed replication strategies are covered by the strategy No Replication, because (i) there is – obviously – no replication functionality and (ii) neither MFRTR nor ECR are meaningful if there a no replicas distributed in the network. Second, there is an implicit interdependency between CC and ECR because of their focus on content classes. This means, CC and ECR are only configured jointly in order to clearly bind content-class-based strategies. Finally,

Strategy Assembly	Get Strategy	Put Strategy	Index Strategy	Storage Strategy	Replacement Strategy	Replication Strategy
DSF4SP_1	Content Class	Content Class	Periodic	Remove Flag	ECR	CC
DSF4SP_2	Parallel Lookup	Transient Replica	Periodic	Transient Replica	MFR	MPP
DSF4SP_3	Parallel Lookup	Transient Replica	Periodic	Remove Flag	MFRTR	MPP
DSF4SP_4	Content Class	Content Class	Periodic	Remove Flag	ECR	MPP, CC
DSF4SP_5	Parallel Lookup	Transient Replica	Periodic	Transient Replica	MFR	PA
DSF4SP_6	Parallel Lookup	Transient Replica	Periodic	Remove Flag	MFRTR	PA
DSF4SP_7	Content Class	Content Class	Periodic	Remove Flag	ECR	PA, CC
DSF4SP_8	Parallel Lookup	Transient Replica	Neighbour	Transient Replica	MFR	CPA
DSF4SP_9	Parallel Lookup	Transient Replica	Neighbour	Remove Flag	MFRTR	CPA
DSF4SP_10	Content Class	Content Class	Neighbour	Remove Flag	ECR	CPA, CC
MDCDN	MDCDN	MDCDN	MDCDN	Basic	MDCDN	MDCDN
Caching	Caching	Basic	Basic	Basic	MFR	--
No Replication	Basic	Basic	Basic	Basic	MFR	--

Table 5.19: Simulation Configuration: Overview of Assembled Strategies

note that the configurations DSF4SP_2, DSF4SP_5, and DSF4SP_8 do not make use the concept of lazy removal of transient replica defined in Section 4.1.2.3, because they rely on the basic replacement strategy MFR.

5.4.1.2 Parameter Configuration of the Proposed Content Placement Strategies

The 10 combinations of the distributed storage service are each simulated with up to 4 different parameter configurations that are presented in Table 5.20. The parameter setting of these configurations is the result of a multitude of simulation runs with varying configurations. They represent the configurations most suitable for analysing the impact of the proposed strategies on query efficiency and content availability.

In addition to the different parameter settings, the table lists the corresponding identifiers of the proposed strategies to ease association with the description of the latter presented in Chapter 4. Moreover, since not all parameters are required / used by all strategy assemblies, the column Strategy Assembly refers to the configurations of the distributed storage service affected by each parameter configuration. Finally, note that while DSF4SP_2 to DSF4SP_10 make use of all four parameter configurations, DSF4SP_1 is simulated with the configurations 3 and 4, only.

Configuration of the Parameters τ, α, and β. Looking at the actual parameter configuration, one can take τ, the schedule look ahead length of the workflow-based replication strategies, as the leading parameter. τ directly affects the two properties α, the maximum number of activities taken into account after XOR transitions both during workflow analysis and replication scheduling, and β, the overall number of activities to schedule for all but the most probable path. Hence, the 4 parameter configurations can

Parameter Description	Parameter	Strategy Assembly	DSF4SP_*_1	DSF4SP_*_2	DSF4SP_*_3	DSF4SP_*_4
Prerequisite - periodic index	t_{INDEX}	{1, ..., 10}	5,000	5,000	5,000	5,000
Prerequisite - period to send periodic index to k neighbours	$t_{INDEX\text{-}NEIGHBOUR}$	{8, 9, 10}	500	500	500	500
Prerequisite - periodic path exeuction history period length	$t_{WORKFLOW}$	{2, ..., 10}	8,000	8,000	8,000	8,000
Prerequisite - #neighbours to send index to	k	{8, 9, 10}	2	2	2	2
Workflow analysis - path length after XOR transition	α	{2, ..., 10}	1	1	2	2
Workflow analysis - path assessment threshold	γ	{5, 6, 7}	1.33	1.41	1.67	1.67
		{8, 9, 10}	1.55	1.63	2.09	2.14
Workflow analysis - off-board storage factor	λ	{2, ..., 10}	1	1	1	1
Workflow analysis - on-board storage factor	σ	{2, ..., 10}	3	3	3	3
Workflow analysis - neighbour interest factor	$I_d{}^N$	{8, 9, 10}	2	2	2	2
Replication scheduling - schedule look ahead	τ	{2, ..., 10}	3	3	4	4
Replication scheduling - #activities to schedule for non-most-propable paths	β	{2, ..., 10}	2	2	3	3
Content replication - max. parallel requests	ζ	{2, ..., 10}	7	7	9	9
Content replication - transient replica TTL interval 1	--	{1, ..., 10}	120	160	180	240
Content replication - transient replica TTL interval 2	--	{1, ..., 10}	180	240	270	360
Content replication - transient replica TTL interval 3	--	{1, ..., 10}	240	320	360	480
Content class availability - #replicas leaf set	Ω	{1, 4, 7, 10}	2	4	2	4
Content class query efficiency - #replicas leaf set	Ψ	{1, 4, 7, 10}	1	2	1	2
Content class query efficiency - #replicas neighbour set	Φ	{1, 4, 7, 10}	1	2	1	2
#GET requests to trigger reactive replication	Γ	{1, 4, 7, 10}	6	3	6	3
Replacement (ECR) - default benefit	Θ	{1}	--	--	0.48	0.48
		{4}	0.48	0.48	0.5	0.5
		{7}	0.48	0.48	0.46	0.46
		{10}	0.49	0.49	0.48	0.48

Table 5.20: Simulation Configuration: Configuration of the Proposed Content Placement Strategies

be distinguished into the rather *risk-averse configuration* with $\tau = 3$, $\alpha = 1$, and $\beta = 2$ (properties 1 and 2) and the more *optimistic configuration* with $\tau = 4$, $\alpha = 2$, and $\beta = 3$ (properties 3 and 4).

Configuration of the Parameter ζ. The configuration of ζ, the maximum number of parallel *GET* requests used in the content replication phase, is calculated as a function of τ shown in Eq. 16. With an average number of content objects needed per activity of 2.37 given the workflows of the evaluation scenario presented in Section 5.2.5, this leads to the values 7 and 9 shown in Table 5.20.

$$\zeta(\tau) = \lfloor \tau \times \text{avg. \#content needed per activity} \rfloor \tag{16}$$

Configuration of the TTL Intervals of Transient Replicas. Also, the TTL intervals of transient replicas are configured subject to the setup of τ. According to the re-scheduling mechanism, the workflow-based replication strategies pre-replicate content for a maximum number of $\tau - 1$ upcoming activities. Note that this value is decremented by 1 to take into account the active activity and solely cover future demand. Moreover, as stated in Section 5.2.5, the average activity processing time is configured to a value of 60 simulation steps; the maximum processing time is set to a value of 80 simulation steps (see Section 5.4.1.4).

As an example, image a simple workflow consisting of the three activities (A1, A2, A3). Assuming a configuration of $\tau = 3$, all activities are covered in the initial schedule. Depending on the total number of content needed by the three activities, it might be possible that all content needs are pre-replicated at the beginning of the processing of activity A1 at simulation time $t = s$. Given the above-stated activity processing times, activity A2 will start at simulation step $t = s+60$ on average and at $t = s+80$ the latest. Activity A3 will start at $t = s + 120$ on average and at $t = s + 160$ the latest.

Consequently, given a value of $\tau = 3$ and an average activity processing time of 60 simulation steps, transient replicas being pre-replicated for future demand should at least be available for $(\tau - 1) \times 60 = 120$ simulation steps. The more optimistic configuration (in terms of replication degree) uses the maximum activity processing time as basis and leads to a value of $(\tau - 1) \times 80 = 160$ simulation steps. The subsequent TTL intervals are configured by simply adding another activity processing time of either 60 or 80 simulation steps depending on the setup of the first interval.

Configuration of the Parameter γ. The configuration of the path assessment threshold γ used by PA and CPA approximates the arithmetic mean of the $P98$ percentile, which was determined based on analyses of test runs with the different configurations shown in Table 5.20. Note that the configurations in the groups (DSF4SP_5, DSF4SP_6, DSF4SP_7) and (DSF4SP_8, DSF4SP_9, DSF4SP_10) use identical values of the parameter γ. This is because γ only relates to the workflow-based replication strategies PA and CPA and should not take into account any state changes caused by the strategies configured in (DSF4SP_6, DSF4SP_7) and (DSF4SP_9, DSF4SP_10), respectively.

Configuration of the Replication Strategy CC. Also, the parameters of the content-class-based replication strategy, i.e., the number of replicas to be placed in the leaf set (Ω, Ψ) or the neighbour set (Φ) as well as the threshold for triggering reactive replication (Γ), are grouped into a rather risk-averse and a more optimistic configuration (again in terms of the replication degree). Thus, the *risk-averse configuration* places a number of $\Omega = 2$ and $\Psi = 1$ replicas in the leaf set of the provider for content of content class *CC_AVAILABILITY* and *CC_QUERY_EFFICIENCY*, respectively. The number of replicas to be placed in the neighbour set is configured to a value of $\Phi = 1$. In contrast, the *optimistic configuration* (properties 2 and 4) doubles all of these values ($\Omega = 4$, $\Psi = \Phi = 2$). Similarly, the risk-averse configuration has a higher boundary for triggering reactive replication $\Gamma = 6$ that is twice as high as the boundary configured for the optimistic configuration $\Gamma = 3$.

Configuration of the Replacement Strategy ECR. The approach used for the configuration of the parameter γ is also applied to the calculation of the default benefit of the replacement strategy ECR (see Section 4.3.2). Note that, even though the benefit calculation used by ECR solely depends on the *GET* request rate that is equal for the different configurations, the setup of the default benefit is tailored to the strategy assemblies DSF4SP_1, DSF4SP_4, DSF4SP_7, and DSF4SP_10 as well as for the related parameter settings (1, 2) and (3, 4). This enables taking into account the impact of the

Parameter Description	MDCDN_1	MDCDN_2
Demand forecast - double exponential smoothing - smoothing factor	0.2	0.2
Demand forecast - double exponential smoothing - forecast periods	7	7
Indirect request rate - upper bound (avg. P98)	3.3	3.5
Periodic index	1,500	2,000
Reconfiguration period	3,000	4,000
Default maintenance effort (reduce number of replicas in network)	3	4

Table 5.21: Simulation Configuration: Configuration of MDCDN

different replication strategies combined with ECR as well as any side effects resulting from the configuration of the workflow-based replication strategies of the two groups (1, 2) and (3, 4) as described afore.

Finally, the parameter e_d of ECR is configured in pre-defined intervals to limit the impact of the parameter on the overall cost calculation. Thus, e_d is set to a value of $e_d = 1$ if content d was accessed during the last 100 simulation steps. If content d was last accessed during the last 250 simulation steps but not during the last 100 simulation steps, e_d is set to a value of $e_d = 2$. For d being accessed in the simulation step interval (250, 500], e_d is set to a value of $e_d = 3$. For all other cases, e_d is set to a value of $e_d = 4$.

5.4.1.3 Configuration of Related Work

MDCDN. The configuration of MDCDN, which is used as representative for active replication strategies (see Section 5.3.6.1), is presented in Table 5.21. The two properties smoothing factor and forecast periods of the statistical demand forecasting method Double Exponential Smoothing are both configured as described in [AMAM04]. Next, the period length used to determine periodic indices as well as the reconfiguration period are configured in a way to again enable evaluation of a rather *risk-averse* (MDCDN_2) and a more *optimistic* configuration (MDCDN_1).

The default maintenance effort explained in Section 5.3.6.1 is defined according to the setup of the reconfiguration period. Finally, the threshold used by the algorithm to address future indirect demand (see *load* parameter in Algorithm 5.6) is set according to the arithmetic mean of the P98 percentile, which was determined based on an analysis of test runs with the varying configurations.

Caching. The reactive strategy Caching explained in Section 5.3.6.2 requires configuration of the threshold Γ used for the creation of local replicas. The strategy is simulated and evaluated in the three configurations CACHING_1, CACHING_2, and CACHING_3 that set this threshold to a value of $\Gamma = 2$, $\Gamma = 4$, and $\Gamma = 0$, respectively.

5.4.1.4 Configuration of the Workflow Engine

The configuration of the workflow engine presented in Section 5.3.5 assigns each node capable of executing workflows a pool of a pre-defined number of workflows. Given the evaluation scenario described in Section 5.2.5, each nomadic device is assigned 3 out of the 4 workflows depicted in Figure 5.17. The actual determination of these 3 workflows follows a normalised Gaussian distribution with mean and variance configured as $\mu = 15$ and $\sigma^2 = 5$, respectively. The selection of workflows is made during simulation execution using a finite range distribution in the interval (0, 1]. This distribution and selection approach is moreover applied for (i) the selection of a workflow out of the workflow pool in the course of a *WORKFLOW* event and (ii) the configuration of XOR transitions (see Section 5.3.5).

Each activity is assigned a minimum activity processing time of 40 simulation steps and a maximum processing time of 80 simulation steps. Based on a multitude of simulation runs with varying configurations, this configuration has been shown to approximate an average activity processing time of 60 simulation steps as defined in Section 5.2.5. Finally, the verification and adjustment of activity-related content needs presented in Section 5.3.5 makes use of the Zipf-like distribution used for modelling content access popularity (see Table 5.13).

5.4.1.5 Message Property Configuration

The configuration of the messages used by the distributed storage service is presented in Table 5.22. This includes the maximum number of overlay hops (i.e., Hops To Live (HTL)) as well as the timeout (i.e., the maximum message delivery time). The HTL is determined based on the expected message routing. For example, while a *GET* message is routed from requester to provider with an assumed maximum number of 8 overlay hops, a *GET RESPONSE* message is always sent from provider to requester based on a single overlay hop. In contrast, the message timeout value is configured as a function of the maximum number of hops as presented in Eq. 17.

The timeout value $f_m(h_m)$ of a message m is calculate based on the maximum number of overlay hops configured for this message h_m multiplied with two times the average delivery time per overlay hop of a *GET* message t_h. This parameter t_h is set to a value of 12 simulation steps according to measurements of multiple simulation runs with different configurations. Moreover, to take into consideration longer delivery times of messages carrying a single or multiple content objects, the above product is multiplied with the weighting factor c_m. Since the *GET* message used to determine the average delivery time per overlay hop does not include content, the weighting factor is set to a value of $c_m = 1$ for messages that do not carry content. For messages carrying a single or multiple content objects the weighting factor is configured as $c_m = 1.5$ and $c_m = 3$, respectively.

$$f_m(h_m) = \lceil h_m \times t_h \times c_m + t_b \rceil \quad (17)$$

h_m : maximum number of overlay hops of message m
t_h : 2 × average delivery time per overlay hop (*GET* message)
c_m : weighting factor subject to content carried by message m
t_b : delivery buffer

The product $h_m \times t_h \times c_m$ is added a delivery time buffer of 15 simulation steps to cover variance in message delivery time and to avoid too many messages from being dropped. This sum is rounded up to eventually come up with the timeout value $f_m(h_m)$.

5.4.1.6 Configuration of the Overlay Network Implementation Pastry

The implementation of the P2P content location and routing substrate Pastry makes use of the following configuration of the parameters defined in [RD01a]. Given an initial number of 2,720 nodes in the evaluation scenario presented in Section 5.2.5 and a measured average number of nodes approximating a value of 2,540 (see Section 5.4.2.1), the parameter N (i.e., nodes in the network) of Pastry is set to a value of 3,000. The parameter b is configured with its typical value of 4. This leads to Pastry Ids being represented with base $2^b = 16$ as hexadecimal numbers. Moreover, the routing table per node is realised as a matrix with $\lceil log_{2^b}(N) \rceil = 3$ rows and $2^b - 1 = 15$ columns hence consisting of a maximum number of 45 nodes. The number of nodes in the leaf and neighbour set is configured to a value of $2^b = 16$. This results in a theoretical number of $log_{2^b}(N)$ overlay hops required for routing messages from source to destination.[57]

In order to prove the validity of the Pastry implementation, the above-stated setup was analysed in detail based on multiple simulation runs with varying configurations and was compared with the implementation of the overlay network Chord [SMK+01] shipped with the simulation framework PlanetSim (remember that Chord could not be used due to the lack of neighbour node support).[58] This comparison illustrated – as expected – that the overlay network implementation Pastry slightly outperforms Chord in terms of the average number of overlay hops required for routing messages from source to destination (remember that Chord requires a theoretical number of $log(N)$ overlay hops). Moreover, to compare not only the efficiency but also the quality of the

[57] Note that the intervals for pinging the successor of a node, each entry of the routing table, and each entry of the neighbour set are configured with constant values of 120, 240, and 240 simulation steps, respectively.

[58] An octal-based configuration with $b = 3$ was analysed as well but led to worse results.

Message Type	Hops	Timeout
FAIL	1	51
GET	8	111
GET ERROR	1	27
GET RESPONSE	1	33
INDEX INVALIDATION	1	27
INIT STORAGE	1	27
INIT STORAGE RESPONSE	1	51
INIT STORAGE ACK	1	27
LEAVE NETWORK	1	51
NEIGHBOUR INDEX	1	27
PUT	8	159
PUT ERROR	1	27
PUT RESPONSE	1	27
REPLACE	2	51
REPLACE ERROR	1	27
REPLACE RESPONSE	1	27
REPLICATE	8	159
REPLICATE PING	1	27
REPLICATE PING RESPONSE	1	27

Table 5.22: Simulation Configuration: Configuration of Message Hop and Timeout Properties of Messages used by the Distributed Storage Service

overlay network implementation, Pastry and Chord were compared in terms of hit rate. Again, both implementations showed very close results.

5.4.2 Analysis of Simulation Data

This section analyses the simulation data given the quality attributes described in Section 5.2.4 and presents the actual evaluation results. Note that the subsequent sections explain the main results of the simulation-based evaluation study and depict a subset of the corresponding charts, only. The complete simulation data is presented in Annex A.4.[59]

The section is structured as follows. Section 5.4.2.1 presents the results that are independent of any content placement strategy and apply to all simulated configurations. The subsequent sections capture the impact of the novel content placement strategies, which are clustered into four groups.

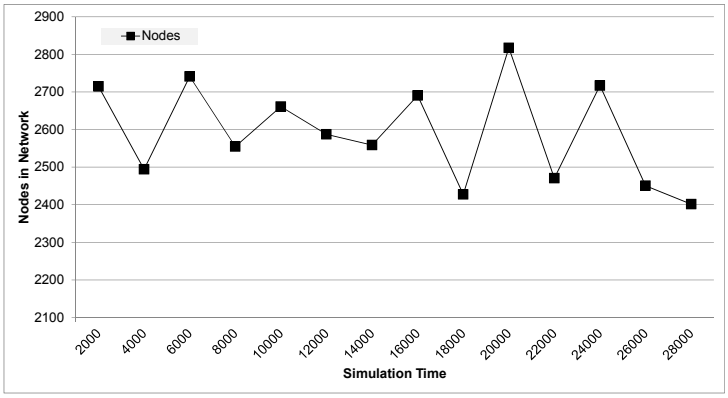
[59] In the analysis, all relative deviations are calculated using the equation $\Delta\% = \frac{a-b}{b}$, with a and b representing the value to compare and the basis to which to compare, respectively.

- Section 5.4.2.2 encompasses the workflow-based replication strategies MPP, PA, and CPA in their configurations DSF4SP_2, DSF4SP_5, and DSF4SP_8, respectively. These configurations do not include any of the proposed replacement strategies or the content-class-based replication strategy, and enable an exclusive analysis of the workflow-based replication strategies.
- In Section 5.4.2.3, the three workflow-based replication strategies are each combined with the proposed replacement strategy MFRTR to capture the effect of the lazy removal properties of transient replicas. This includes the three configuration DSF4SP_3, DSF4SP_6, and DSF4SP_9.
- Section 5.4.2.4 extends the analysis by taking into account the content-class-based content placement strategies CC and ECR. This group consists of the configuration DSF4SP_1 to exclusively evaluate the effect of the two content-class-based content placement strategies as well as the three configurations DSF4SP_4, DSF4SP_7, and DSF4SP_10 that capture the complete spectrum of the proposed strategies taking into account workflows, content classes, and transient replicas with lazy removal properties.
- Section 5.4.2.5 presents an analysis of the related work MDCDN, Caching, and No Replication as well as a comparison of the latter with the proposed content placement strategies.

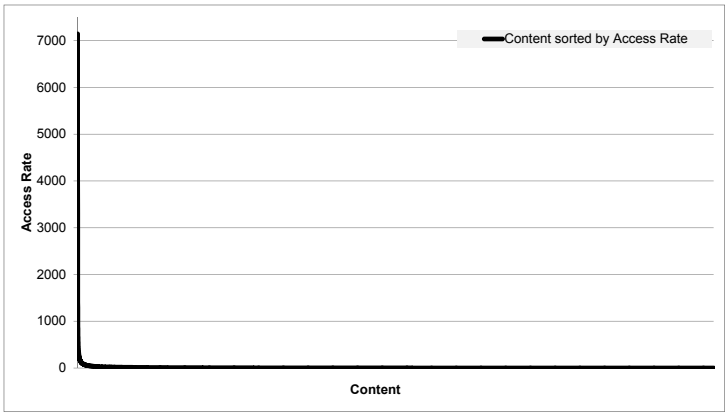
Each of the above-stated sections consist of a conceptual analysis of the expected results and a comparison between the latter and the actual simulation data. This addresses the potential issue of synthetic data used within the simulation framework and allows for a more detailed evaluation study.

5.4.2.1 Configuration-independent Evaluation Results

The modelling of the evaluation scenario presented in Section 5.2.5 resulted in an overall number of 31,823 overlay network events being made up of 17,090 *JOIN* events, 13,244 *LEAVE* events, and 1,489 *FAIL* events. Given the initial setup of the simulated network with 2,720 nodes, this led to the distribution of nodes in the network (*QA1*) presented in Figure 5.32(a). While there is a moderate node churn rate in the simulation time interval $[0; 18,000]$, the increasing variation of nodes in the network after simulation step 18,000 results from the configuration of *LEAVE* and *FAIL* events of nomadic devices (900 nodes on average). Both events follow a Weibull distribution (see Section 5.2.3.1) with shape and scale being set to values of 15 and 28,800, respectively (see Table 5.15). Since nomadic devices are equipped with relatively large storage capacity, this drop may have an impact on both content availability and query efficiency. In addition, each simulation configuration processed a total number of 215,664 overlay service events, which are comprised of 102,728 *PUT* events, 107,002 *GET* events, and 5,934 *WORKFLOW* execution events.



(a)



(b)

Figure 5.32: Configuration-independent Evaluation Results: (a) nodes in network, (b) content popularity

Figure 5.32(b) presents the actual content access popularity (*QA3*) averaged over all simulation runs. This curve clearly represents the Zipf distribution specified as part of the modelled evaluation scenario (see Table 5.16) and leads to a median number of 2 requests per content object.

5.4.2.2 Workflow-based Replication Strategies

This section presents an analysis of the workflow-based replication strategies MPP, PA, and CPA in their configurations DSF4SP_2, DSF4SP_5, and DSF4SP_8, respectively. These configurations do not include any of the proposed replacement strategies or

the content-class-based replication strategy, and enable an exclusive assessment of the workflow-based replication strategies.

Expected Results

Properties of Replicas and Transient Replicas. As explained in Section 4.1, both PA and CPA extend MPP by pre-replicating activity-related content needs for alternative paths other than the path with the highest execution probability if the assessment value of these paths exceeds a configurable threshold. Yet, in either case, PA and CPA capture the most probable path covered by MPP. For this reason, it is expected that both PA and CPA increase the transient replica ratio (*QA5*) as well as the average number of transient replicas in the network (*QA2.3*) compared to MPP. PA and CPA can be assumed to produce similar values for *QA5* and *QA2.3*, because of the adapted configuration of the path assessment parameter γ (see Section 5.4.1.2).

CPA should result in an increased transient replica utilisation (*QA6*) compared to PA driven by its incorporation of cooperative knowledge. This should also be reflected in the replica ratio (*QA4*) and the corresponding average number of replicas in the network (*QA2.2*), i.e., the higher the utilisation of transient replicas the higher the number of transient replicas being transformed into "persistent" replicas.

MPP is expected to create the lowest pre-replication failure compared to PA and CPA due its rather pessimistic approach. Moreover, one can assume MPP to result in a lower replica ratio compared to PA and CPA, i.e., the lower the number of transient replicas the lower the probability of generating "persistent" replicas out of transient replicas. However, due to the increased pre-replication wastage of PA and CPA as well as potential replacement operations resulting from the latter, a proportional relation between transient replicas and replicas cannot be assumed in general.

Analysis of Transient Replica Ratios. The impact of PA and CPA on the transient replica ratio can be analysed similar to the approximation of the pre-replication failure presented in Section 4.1.2.4. Thus, the analysis assumes XOR branches to be distributed uniformly and assesses the effect of different configurations of the parameters τ, α, and β. Also, to facilitate the approximation of average values, the execution of workflows is supposed to follow a uniform distribution. The simulation data used for calculating the path assessment threshold γ (see Section 5.4.1.2) revealed that an average portion of 33% of paths other than the most probable path are handled by the multi-path pre-replication concept of PA and CPA. This factor is taken into consideration for approximating the expected increase of the transient replica ratio.

An exemplary calculation is depicted in Figure 5.33 for the workflow WF1. The configuration $(\tau, \alpha, \beta) = (3, 1, 2)$ results in an equal transient replica ratio for all three proposed workflow-based replication strategies. This is because of the parameter τ limiting the initial schedule to include activities of the common sub path and the most probable path, only (e.g., $S_S(S, 1, 2A)$). Also, there is no extension of the initial sched-

Workflow	WF1	
Configuration	tau	3
	alpha	1
	beta	2
Avg. #content per path		11.25
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	0.00
	probability	33%
	actual	0.00

Delta transient replica ratio	0.00%

Comment
* tau limits activity to be pre-replicated after XOR to most probable path
* there is no re-scheduling because schedule is cleared after XOR

Workflow	WF1	
Configuration	tau	4
	alpha	2
	beta	3
Avg. #content per path		11.25
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	2.75
	probability	33%
	actual	0.91

Delta transient replica ratio	8.07%

Comment
* tau limits additional activity to be pre-replicated to a single activity

XOR	Prob	MPP	Add. Paths	Add. #Content	Avg. Add. #Content
1	1	2A	2B	4	
			2C	4	
			2D	1	3.00
	1	2B	2A	2	
			2C	4	
			2D	1	2.33
	1	2C	2A	2	
			2B	4	
			2D	1	2.33
	1	2D	2A	2	
			2B	4	
			2C	4	3.33
					2.75

Figure 5.33: Approximation of the Transient Replica Ratio Increase of PA / CPA over MPP for Workflow WF1

ule, since it is cleared and built from scratch after processing the XOR transition. This changes in case of the parameters being set to the values $(\tau, \alpha, \beta) = (4, 2, 3)$. This way, the initial schedule consists of (i) activities belonging to the common sub path, (ii) one activity of the most probable path, as well as (iii) one activity of another but the most probable path (if any). The approximation calculates the average number of content pre-replicated for this potential additional activity taking into account all possible combinations of (ii) and (iii). The resulting additional number of pre-replicated content (2.75 content objects) is multiplied with the above-stated probability of pre-replicating content needs of activities of paths other than the most probable path (33%). This value is put in relation to the average number of content needed per path of the workflow, which approximates the number of content objects pre-replicated by MPP (11.25 content objects). Eventually, this yields the approximated increase of the transient replica ratio of PA and CPA compared to MPP (8.07%).

An analysis of the workflows used in the evaluation scenario (see Figure 5.17), revealed an approximated increase of the transient replica ratio from MPP to PA and CPA of approximately 6.2%. Note that this approach can also be used for estimating the increase of the average number of transient replicas in the network. The complete calculation covering the workflows WF2, WF3, and WF4 is presented in Annex A.4.3.

Query Efficiency and Content Availability. In terms of query efficiency of workflow-related content requests (*QA8*), one can expect PA and CPA to improve MPP. This is

ID	Quality Attribute	Assumption
A1	QA2.2	The number of replicas in the network increases with the number and the utilisation of transient replicas.
A2	QA2.3	PA and CPA lead to a higher number of transient replicas in the network than MPP.
A3	QA2.3	PA and CPA lead to a similar number of transient replicas in the network.
A4	QA5	PA and CPA lead to a higher transient replica ratio than MPP (approx. 6.2%).
A5	QA5	PA and CPA lead to a similar transient replica ratio.
A6	QA6	CPA leads to an enhanced transient replica utilisation compared to PA.
A7	QA6	MPP leads to a lower pre-replication failure than PA and CPA.
A8	QA7	PA and CPA lead to an improved overall query efficiency compared to MPP (limited effect).
A9	QA7	PA and CPA lead to a similar overall query efficiency.
A10	QA8	PA and CPA lead to an improved workflow-related query efficiency compared to MPP.
A11	QA8	PA and CPA lead to a similar workflow-related query efficiency.
A12	QA13	PA and CPA lead to an enhanced content availability than MPP (limited effect).

Table 5.23: Simulation Analysis: Expected Results

reasoned by the more optimistic approach of PA and CPA as well as the resulting increased probability of pre-replicating content needs of the actually processed activities. Also, this points out again the trade-off between pre-replication wastage and enhancement of query efficiency. The impact of PA and CPA on *QA8* should not differ significantly because of the adapted path assessment threshold configuration.

Regarding the overall effect on query efficiency (*QA7*), it has to be taken into consideration that the three proposed replication strategies MPP, PA, and CPA target optimisation of query efficiency of workflow-related content needs, only, and that the configurations DSF4SP_2, DSF4SP_5, and DSF4SP_8 do not include any additional replication functionality. Hence, it can be assumed that the impact of the three strategies on *QA8* also applies to *QA7*. However, given that only 45% of the total number of *GET* requests result from workflow-related content needs, the effect should decrease noticeably.

Similar to *QA7*, there is also only an indirect impact of the three replication strategies on content availability (*QA13*). The strategies may enhance availability of activity-related content needs. For this reason – if at all – one may experience a slight increase of the overall content availability with PA and CPA having a higher effect due to their more optimistic nature.[60] The expected results are summarised in Table 5.23.

Expected Effect of the Configuration Options

As stated in Section 5.4.1.2, each assembly of the proposed content placement strategies (DSF4SP_1..10) is simulated with four different parameter configurations. The two configurations 3 and 4 both include an increased setup of the schedule look ahead length τ compared to the configurations 1 and 2. This setting directly affects the configuration

[60] Note that the configurations DSF4SP_2, DSF4SP_5, and DSF4SP_8 do not contain any content-class-related functionality. Hence the quality attributes *QA9* and *QA14* are not applicable.

of the parameters α, β, and ζ, and leads to more optimistic pre-replication. Moreover, the TTL intervals of transient replicas are set based on the average activity processing time for the configurations 1 and 3, and the maximum activity processing time for the configurations 2 and 4. Hence, transient replicas being created with configurations 1 and 3 have a higher probability of being removed compared to transient replicas being created with configurations 2 and 4.

Quality Attributes Directly Affected by the Configuration Options. As a result, for each strategy assembly except DSF4SP_1[61], it can be assumed that the transient replica ratio (*QA5*) is increased by the configurations 3 and 4 compared to the configurations 1 and 2. The average number of transient replicas in the network (*QA2.3*) is directly affected by the length of the TTL intervals. For this reason, it is expected that the configurations 2 and 4 lead to an increased average number of transient replicas compared to the configurations 1 and 3, respectively. Following the same argumentation, on the one hand, the configurations 1 and 2 are expected to result in lower pre-replication wastage (*QA6*) compared to the configurations 3 and 4 (lower value of τ). On the other hand, the configurations 1 and 3 can be assumed to lead to higher pre-replication wastage than the configurations 2 and 4, respectively (shorter TTL intervals). This also applies to query efficiency of workflow-related content needs (*QA8*) and – indirectly – to the overall query efficiency (*QA7*). While the configurations 3 and 4 are expected to enhance query efficiency compared to the configurations 1 and 2, the shorter TTL intervals of the configurations 1 and 3 are likely to worsen query efficiency compared to the configurations 2 and 4, respectively.

The pre-replication wastage of MPP ca be approximated following the procedure explained in Section 4.1.2.4. Table 5.24 summarises the expected pre-replication failure in the average and worst case for different configurations of τ and α. Details about the calculation are presented in Annex A.4.2. Based on the assumption of all workflows being executed with equal probability, the configurations 1 and 2 have expected pre-replication failures of 16.38% and 40.72% in the average and worst case, respectively. Similarly, the configurations 3 and 4 have expected values of 25.89% and 65.74%.

Quality Attributes Indirectly Affected by the Configuration Options. As stated afore, the replica ratio (*QA4*) cannot be directly derived from the transient replica ratio. Also, since content availability (*QA13*) is not addressed explicitly by the workflow-based replication strategies, the different parameters settings of the configurations 1 to 4 are expected to have a minor effect, only. One can assume the higher transient replica ratios of the more optimistic approaches 3 and 4 to enhance content availability compared to the configurations 1 and 2, and the shorter TTL intervals of the configurations 1 and 3 to worsen content availability compared to the configurations 2 and 4, respectively. The expected effect of the configuration options is summarised in Table 5.25.

[61] DSF4SP_1 does not apply any workflow-based replication strategy and is not affected by the above-summarised parameter setting.

Workflow	Configuration		Pre-Replication Failure	
	tau	alpha	average	worst case
Workflow 1	3	1	18.33%	35.56%
	4	2	28.33%	71.11%
Workflow 2	3	1	24.69%	55.56%
	4	2	37.04%	77.78%
Workflow 3	3	1	4.04%	18.18%
	4	2	14.65%	54.55%
Workflow 4	3	1	18.44%	53.57%
	4	2	23.56%	59.52%
All Workflows (avg.)	3	1	16.38%	40.72%
All Workflows (avg.)	4	2	25.89%	65.74%

Table 5.24: Simulation Analysis: Approximation of Pre-Replication Failure

ID	Quality Attribute	Assumption
A13	QA2.3	CFG 2 and 4 lead to a higher number of transient replicas in the network than CFG 1 and 3, respectively.
A14	QA5	CFG 3 and 4 lead to a higher transient replica ratio than CFG 1 and 2.
A15	QA6	CFG 1 and 2 lead to a lower pre-replication failure than CFG 3 and 4.
A16	QA6	CFG 1 and 3 lead to a higher pre-replication failure than CFG 2 and 4, respectively.
A17	QA7/8	CFG 3 and 4 lead to an improved query efficiency than CFG 1 and 2.
A18	QA7/8	CFG 1 and 3 lead to a lower query efficiency than CFG 2 and 4, respectively.

Table 5.25: Simulation Analysis: Expected Effect of the Configuration Options

	QA2.2	QA2.3	QA4	QA5	QA6 (0)	QA7	QA8	QA10	QA11	QA13	QA15
DSF4SP_2_1	84.98	297.07	0.27%	0.96%	13.89%	28.50	20.95	31.48	24.35	71.43%	72.67%
DSF4SP_2_2	128.91	387.52	0.41%	1.25%	12.32%	28.21	20.28	31.48	23.70	71.05%	70.90%
DSF4SP_2_3	146.15	561.89	0.47%	1.80%	19.53%	26.21	14.60	29.39	17.41	70.14%	68.86%
DSF4SP_2_4	177.74	732.73	0.57%	2.37%	17.57%	25.82	14.02	29.06	16.71	69.99%	67.79%
DSF4SP_5_1	83.45	298.11	0.27%	0.96%	15.60%	28.80	21.05	31.87	24.46	70.63%	72.00%
DSF4SP_5_2	132.46	393.27	0.43%	1.26%	13.67%	27.91	19.69	31.03	22.79	71.27%	71.09%
DSF4SP_5_3	133.79	582.72	0.43%	1.88%	21.50%	25.99	13.89	28.95	16.40	70.59%	69.44%
DSF4SP_5_4	185.59	755.92	0.60%	2.44%	20.46%	25.75	13.42	28.97	16.05	70.81%	68.54%
DSF4SP_8_1	108.01	291.63	0.35%	0.94%	15.49%	28.39	20.83	31.31	23.96	69.65%	70.94%
DSF4SP_8_2	122.36	392.56	0.39%	1.26%	13.18%	28.28	20.22	31.66	23.78	71.38%	71.26%
DSF4SP_8_3	153.68	579.75	0.49%	1.85%	21.56%	25.95	13.85	29.16	16.57	70.75%	69.13%
DSF4SP_8_4	181.13	752.33	0.58%	2.42%	20.29%	25.71	13.47	29.07	16.09	70.19%	67.94%

Table 5.26: Evaluation Results of the Workflow-based Replication Strategies

Simulation Results

The simulation results are depicted in Table 5.26. The expectations of the effect of the configuration options presented in Table 5.25 are reflected by these results in all cases. Also, as discussed before, there is no evident relation between the configuration options and the replica ratio. While an increasing transient replica ratio typically results in a growth of the replica ratio, the configurations 2 and 3 of the strategy assembly DSF4SP_5 can be used as an example that this relation not always applies. In these cases, the increase of the transient replica ratio has almost no effect on the replica ratio. Similarly, there is no clear relation between the configuration options and their impact on content availability across the three strategy assemblies. The variance in content availability is below 2.5%.

On this basis, a single configuration can be used as for comparing the proposed workflow-based replication strategies. The following analysis uses the configuration 3 as representative and moreover provides the average results of all four configurations for each strategy to enable a more detailed analysis. The data presented in Table 5.26 is illustrated by the Figures 5.34 and 5.35 that capture the quality attributes addressed by the three configurations DSF4SP_2_3, DSF4SP_5_3, and DSF4SP_8_3.

Assumptions A4 and A5. As expected, both PA and CPA increase the transient replica ratio (*QA5*) compared to MPP by 4.6% and 3%, respectively. The average results of all four configurations lead to a reduced increase of 2.7% (PA) and 1.5% (CPA). These values are reasonable given the approximated increase of 6.2% described afore.[62] Moreover, given the adapted setting of the path assessment threshold γ for PA and CPA, their transient replica ratios only differ by 1.5% for configuration 3 and 1.2% on average. Hence, the assumptions *A4* and *A5* are reflected by the simulation results.

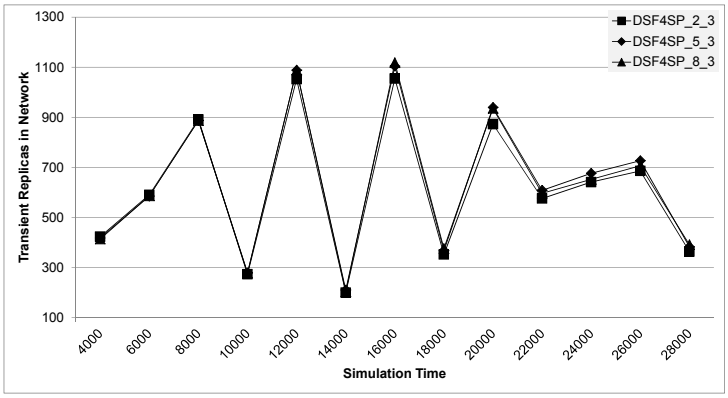
Assumptions A2 and A3. This effect furthermore applies to the average number of transient replicas in the network (*QA2.3*). Compared to MPP, PA and CPA increase this property by 3.7% and 3.2%, respectively. Averaged over all configurations, values of 2.6% for PA and 1.9% for CPA were measured. Consequently, the assumptions *A2* and *A3* are backed up by the simulation results. The minor difference between the results is reflected in Figure 5.35(a) depicting the evolution of the number of transient replicas over time. The strong variation within intervals of approximately 4,000 simulation steps is caused by the configuration of the *WORKFLOW* execution event, which follows a normal distribution with a mean value of 3,600 simulation steps. The drop after simulation step 26,000 reflects the drop in the number of nodes explained in Section 5.4.2.1.

Assumption A7. PA and CPA both more optimistically trade pre-replication wastage for enhancement of query efficiency compared to MPP. This leads to the expected increase in absolute and relative pre-replication failure (*QA6*), which approximate values of 15% and 10%, respectively, both for configuration 3 and the average of all configurations. Thus, the assumption *A7* is confirmed by the simulation results. Interestingly, the absolute difference of the relative pre-replication failure of MPP and PA / CPA is measured with a value around 2%, only, for both cases (see Figure 5.34(c–e)).

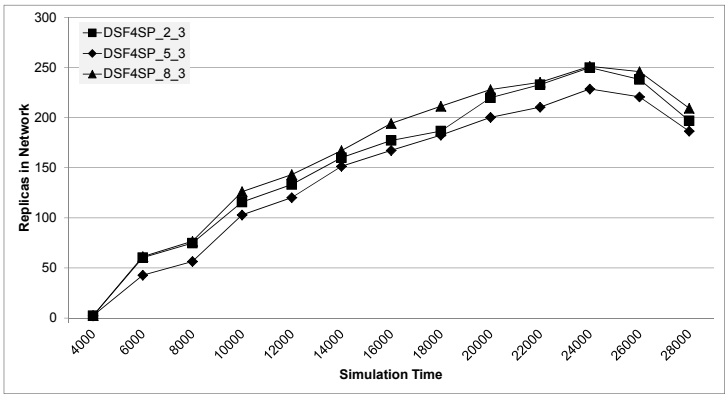
Assumptions A1 and A6. The utilisation of transient replicas is further reflected in the replica ratio (*QA4*) and the average number of replicas in the network (*QA2.2*). Remember that for the given strategy assemblies, transient replicas being transformed into "persistent" replicas represent the only means for creating the latter. CPA increases *QA4* and *QA2.2* of PA by 13.7% and 14.9%, respectively, for configuration 3. The average increase approximates 5% for both properties. Also, CPA results in a growth of the two properties *QA4* and *QA2.2* compared to MPP for both configuration 3 and the averaged configurations (around 5% for all four cases).

While the above-stated relation between CPA and PA meets the expected effect of incorporating cooperative knowledge, the relation does not apply to all configurations and shows significant variations. For example, with respect to *QA2.2*, there is a measured

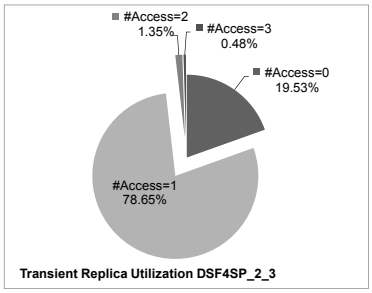
[62] The unexpected decrease in both the transient replica ratio and the average number of transient replicas in the network from DSF4SP_2_1 to DSF4SP_8_1 might result from some minor variations caused by the setup of the workflow engine (remember that the control flow proceeds after 80 simulation steps even if not all activity-related content needs were served). Yet, since the absolute difference for *QA5* is only at a value of 0.02% and the average values support the expected behaviour, this effect is not analysed any further.



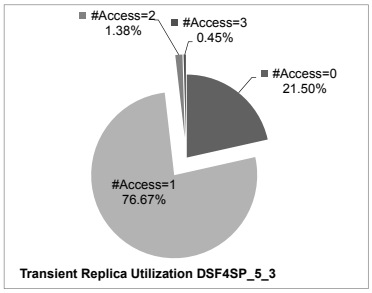
(a)



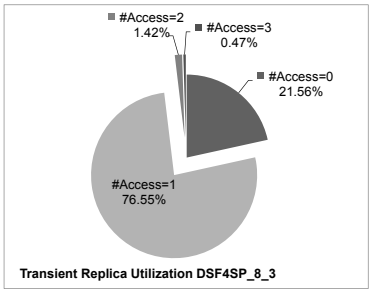
(b)



(c)

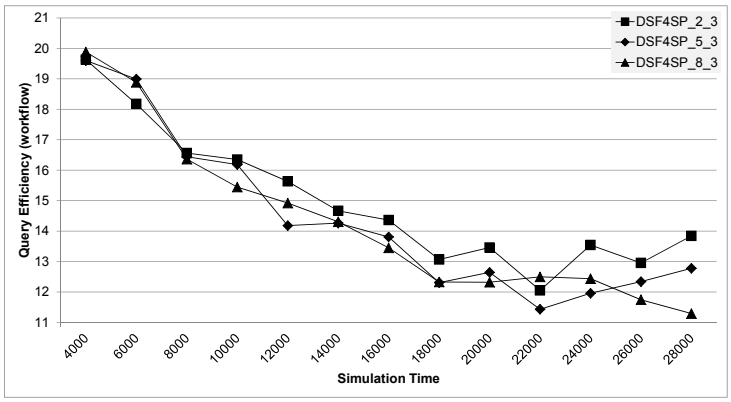


(d)

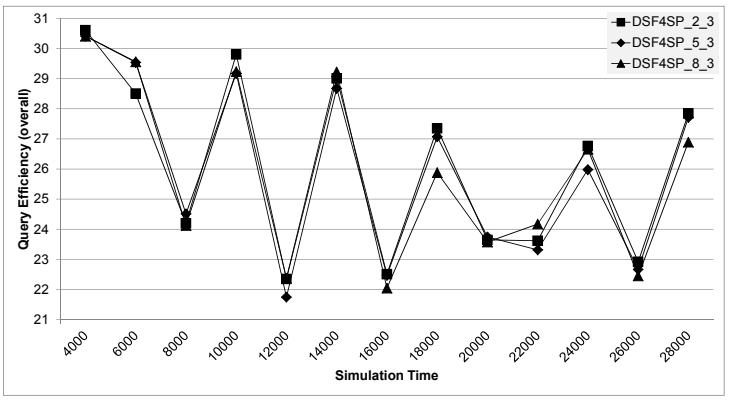


(e)

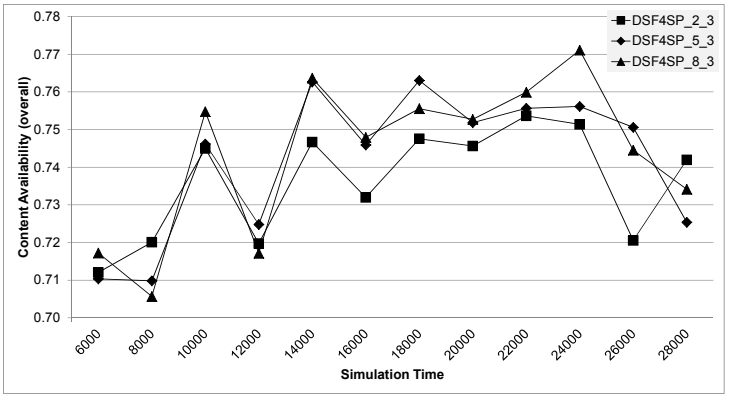
Figure 5.34: Evaluation Results of the Workflow-based Replication Strategies (1/2): (a) transient replica in network, (b) replica in network, (c) transient replica utilisation DSF4SP_2_3, (d) transient replica utilisation DSF4SP_5_3, (e) transient replica utilisation DSF4SP_8_3



(a)



(b)



(c)

Figure 5.35: Evaluation Results of the Workflow-based Replication Strategies 2/2: (a) query efficiency (workflow), (b) query efficiency (overall), (c) content availability (overall)

increase of almost 30% for configuration 1; configuration 2 results in a decrease of more than 7%. Although with less variation, this effect can also be seen for configurations 3 and 4. Similarly, PA worsens MPP for configurations 1 and 3, while it improves the latter for configurations 2 and 4. While the correlated assumptions *A1* and *A6* seem to be confirmed by the average simulation results, a detailed analysis revealed a dependency between the strategies and the TTL interval configuration. Thus, the shorter TTL interval setting of configurations 1 and 3 better fits the strategies MPP and CPA, whereas the longer intervals suite the PA approach.

The evolution of the number of replicas in the network over time is illustrated in Figure 5.34(b). It shows a continuous growth of the number of replicas for all configurations. The drop after simulation step 24,000 again reflects the node distribution explained in Section 5.4.2.1. While the number of transient replicas in the network is subject to significant variations and reaches values above 1,100 objects, the number of replicas increases moderate and remains below 251 objects. This reflects the expected benefit of the concept of transient replicas, which in particular applies to scenario with resource-limited nodes.

Assumption A10. The growth of the properties *QA5* as well as *QA2.3* effected by PA and CPA is further reflected in the query efficiency of workflow-related content requests (*QA8*). Compared to MPP, PA and CPA both improve *QA8* by almost 5% for configuration 3 and more than 2.1% in the average case. Even further, the impact of PA and CPA on *QA8* is slightly stronger than the increase of *QA5* and *QA2.3* both for configuration 3 and the averaged configuration case. This is a very promising result, which confirms the assumption *A10* and demonstrates the benefit of trading higher transient replica ratios and – as a result – higher pre-replication failure for enhancement of query efficiency of workflow-related requests.

Assumption A11. While PA and CPA lead to very close results, they again show the dependency on the TTL interval configuration. Thus, PA improves CPA in terms of *QA8* for configurations 2 and 4; CPA benefits from configurations 1 and 3.

With respect to the relative query efficiency of workflow-related content requests (*QA11*), PA achieves the best results. Thus, for configuration 3, PA outperforms MPP and CPA by 5.8% and 1.1%, respectively. This effect also applies to the average configurations. Hence, given the setup of *QA11* presented in Section 5.2.4.3, PA best balances transient replica ratios and the impact on query efficiency (*local view*).

Overall, as illustrated in Figure 5.35(a), all three workflow-based replication strategies are able of significantly reduce *QA8* in the course of the simulation by almost 40%. This is a very promising result, in particular with regard to long-running smart product systems. Again, the partly increase after simulation step 22,000 results from the node distribution (see Section 5.4.2.1).

Assumptions A8 and A9. As expected, the effect also (indirectly) applies to the overall query efficiency (*QA7*), for which MPP is improved by PA and CPA by 0.3% and 0.4%,

respectively (averaged configurations). This reduced impact results from the distribution of content requests presented in Section 5.4.2.1. The slight increase between PA and CPA is caused by the incorporation of content interest of neighbours, i.e., CPA also increases query efficiency of surrounding smart products (*cooperative view*).

Again, with respect to the relative overall query efficiency (*QA10*), PA is able to marginally outperform MPP and CPA. While this is driven by the enhancement of *QA7* with respect to MPP, PA improves CPA because of its lower replica ratios and the reduced number of replicas in the network.

The evolution of *QA10* over time is illustrated in Figure 5.35(b). While the overall query efficiency is significantly higher than the query efficiency of workflow-related content requests, it follows the same decreasing trend (cp. Figure 5.35(a)). Moreover, the strong variation is in line with the variation seen in Figure 5.34(a). This is indeed reasonable, because the overall query efficiency is only addressed by the optimisation of query efficiency of workflow-related content requests.

Assumption A12. The impact of the three workflow-based replication strategies on both content availability (*QA13*) and relative content availability (*QA15*) is negligible (below 1% for the averaged configurations). These close results are also reflected in Figure 5.35(c). Yet, it is important to note that content availability improves over simulation time by approximately 4.5% even though it is not directly addressed by any of the three strategies. Again, the zig-zag pattern and the drop after simulation step 24,000 reflect the variation of the number of transient replicas and the node distribution, respectively.

Conclusion

In conclusion, both PA and CPA enhance MPP by their more optimistic trading of transient replica ratios as well as pre-replication failure for enhancement of query efficiency of workflow-related content requests. Furthermore, even though PA slightly outperforms CPA in terms of total and relative query efficiency of workflow-related requests, the actual difference is (i) marginal because of the adapted configuration of the threshold parameter γ as well as (ii) subject to the applied configuration. While this solely captures the local view on a node, CPA better addresses the needs of nodes in the environment by taking into consideration cooperative knowledge. This is not only reflected by the slight increase of overall query efficiency but – in particular – by the considerable growth of transient replica utilisation. For the averaged configurations, CPA creates a lower number of transient replicas and has a lower relative pre-replication failure, but leads to an increase of the replica ratio. Given that the utilisation of transient replicas represents the only means for creating "persistent" replicas, this is an important result. Even further, taking into account the modelling of content popularity described in Section 5.2.3, this effect is expected to increase in scenarios with access patterns stronger following geographic locality properties.

The communication overhead of CPA resulting from the exchange of content interest is in fact negligible. For example, given the parameter setting of Pastry presented in Section 5.4.1.6 and an average number of 2,540 nodes in the network, the number of *NEIGHBOUR INDEX* messages exchanged by CPA represent an average portion of less than 1.5% of Pastry's *PING* messages used for maintaining the structured overlay network. As a result, given the broader optimisation of CPA, CPA should be preferred over PA as long as its communication overhead does not represent an issue in the target scenario.

Overall Result
PA most efficiently balances enhancement in query efficiency with (transient) replica ratios from the perspective of nodes targeting workflow execution (local view). Yet, by its cooperative approach, CPA enhances content access for all nodes (cooperative view), while performing only slightly worse than PA for nodes executing workflows. For this reason, if the communication overhead of CPA did not represent a limiting factor, CPA should be preferred over PA. Else, PA should be applied and complemented by a content placement strategy that targets overall enhancement of query efficiency and content availability. Finally, also the simple strategy MPP is capable of significantly improving workflow-related content access. Hence, driven by its risk-averse pre-replication, MPP is the strategy of choice for very resource-limited nodes targeting workflow execution.

5.4.2.3 Impact of Lazy Removal Properties of Transient Replicas

This section captures the effect of the proposed replacement strategy MFRTR that makes use of the lazy removal properties of transient replicas. To enable clear comparison with MPP, PA, CPA assessed in Section 5.4.2.2, the following analysis encompasses the three strategy assemblies DSF4SP_3, DSF4SP_6, and DSF4SP_9. Again, configuration 3 is used as representative and is complemented by information on the results per strategy averaged over all configuration options.

Expected Results

In general, MFRTR is expected to enable a much more effective use of the available storage capacity. Moreover, while MFRTR most likely causes an increase of the number of replacement operations, there should not be a significant growth of the number of exchanged replacement messages (remember that transient replicas flagged to be removed are deleted in case of replacement operations instead of being replaced).

Properties of Replicas and Transient Replicas. The lazy removal properties of transient replicas are expected to increase both the transient replica ratio (*QA5*) as well as

ID	Quality Attribute	Assumption
A19	QA2.2	MFRTR leads to a higher number of replicas in the network.
A20	QA2.3	MFRTR leads to a higher number of transient replicas in the network.
A21	QA4	MFRTR leads to a higher replica ratio.
A22	QA5	MFRTR leads to a higher transient replica ratio.
A23	QA6	MFRTR leads to an enhanced transient replica utilisation.
A24	QA6	MFRTR leads to a reduced pre-replication failure.
A25	QA7	MFRTR leads to an improved overall query efficiency.
A26	QA8	MFRTR leads to an improved workflow-related query efficiency (strongest for MPP).
A27	QA13	MFRTR leads to an enhanced content availability.

Table 5.27: Simulation Analysis: Expected Results

the average number of transient replicas in the network (*QA2.3*). In contrast, the absolute number of created transient replicas is supposed to decrease. This is a side effect of the growth of *QA5* and *QA2.3*, which might obviate the need for creating new transient replicas. Also, the lazy removal properties should improve utilisation of transient replicas (*QA6*). This is reasoned by a simple relationship: the longer a transient replica is accessible, the higher is the probability of the transient replica being accessed. Even further, this is expected to result in a lower overall pre-replication failure as well as an increase of the replica ratio (*QA4*) and the average number of replicas (*QA2.2*).

Query Efficiency. Also, because of the increased probability of content required during workflow execution being available on-board, query efficiency of workflow-related content requests (*QA8*) is supposed to be improved by MFRTR. This effect is expected to be strongest for MPP due to its pessimistic pre-replication approach. The impact on *QA11*, the relative query efficiency of workflow-related content requests, depends on the relation between the increase of *QA2.3* and *QA2.2* on the one hand, and the enhancement of *QA8* on the other hand. This effect also applies to the overall (relative) query efficiency. While one can assume that the increasing number of (transient) replicas in the network positively affects *QA7*, the impact on relative query efficiency (*QA10*) has to be studied using the simulation data.

Content Availability. Even though the simulation model does not capture network partitioning, the TTL and HTL properties of messages affect content availability (see Section 5.4.1.5). For this reason, higher replication degrees are likely to improve content availability (*QA13*). Again, the impact of MFRTR on relative content availability (*QA15*) depends on the relation between the increase of *QA2.3* and *QA2.2*, and the enhancement of *QA13*. The expected results are summarised in Table 5.27.

Simulation Results

The simulation data of the strategy assemblies DSF4SP_3_3, DSF4SP_6_3, and DSF4SP_9_3 are summarised in Table 5.28 and illustrated by Figures 5.36 and 5.37.

	QA2.2	QA2.3	QA4	QA5	QA6 (0)	QA7	QA8	QA10	QA11	QA13	QA15
DSF4SP_3_1	93.48	10,842.94	0.30%	34.83%	12.15%	25.95	15.36	35.23	22.04	63.52%	55.65%
DSF4SP_3_2	118.37	10,791.71	0.38%	34.82%	10.06%	25.58	15.02	34.35	21.28	62.44%	54.62%
DSF4SP_3_3	129.66	14,092.09	0.42%	45.31%	17.95%	23.50	9.09	32.01	12.99	62.00%	52.94%
DSF4SP_3_4	175.84	14,367.19	0.56%	46.15%	16.43%	23.59	9.25	32.24	13.31	63.02%	53.35%
DSF4SP_6_1	92.63	10,861.09	0.30%	35.07%	13.56%	25.96	15.48	34.49	21.92	62.30%	54.92%
DSF4SP_6_2	127.61	10,908.22	0.41%	35.16%	11.49%	25.63	14.82	34.30	20.65	62.27%	54.45%
DSF4SP_6_3	141.75	14,724.21	0.45%	47.02%	20.55%	23.25	8.64	31.53	12.34	63.08%	53.51%
DSF4SP_6_4	184.56	14,450.42	0.60%	46.59%	18.90%	23.53	8.74	32.24	12.54	62.29%	52.85%
DSF4SP_9_1	90.06	11,045.75	0.29%	35.23%	14.06%	25.31	14.75	33.73	20.80	63.06%	55.20%
DSF4SP_9_2	119.08	10,869.61	0.38%	34.87%	11.31%	25.76	15.10	34.61	21.50	62.51%	54.67%
DSF4SP_9_3	136.52	14,293.61	0.44%	45.78%	20.75%	23.40	8.89	32.09	12.87	62.41%	53.02%
DSF4SP_9_4	187.57	14,660.83	0.60%	47.00%	18.59%	23.09	8.47	31.71	12.18	62.63%	52.85%

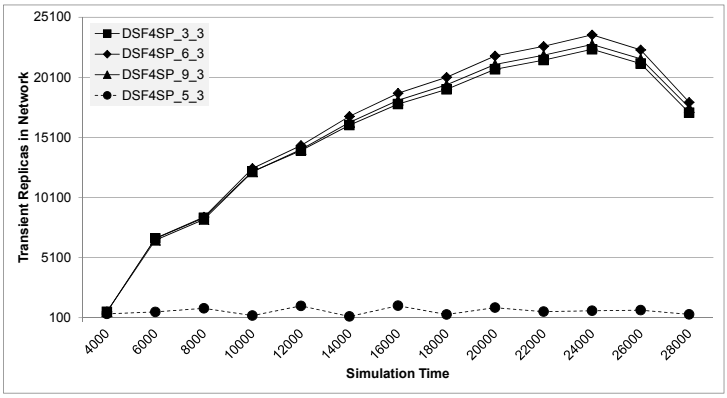
Table 5.28: Evaluation Results of the Replacement Strategy MFRTR

For reasons of comparability, each of these figures includes the best result of the "plain" strategies DSF4SP_2_3, DSF4SP_5_3, and DSF4SP_8_3 for each quality attribute.

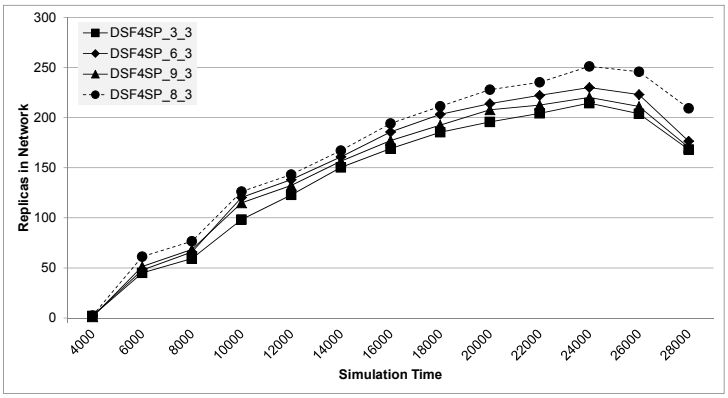
Assumptions A20 and A22. The incorporation of lazy removal properties of transient replicas leads to a tremendous decrease of the absolute number of generated transient replicas. This consistently applies to all three workflow-based replication strategies and yields an average decrease of approximately 55% for both configuration 3 and the averaged configurations (see Table A.32). Still, configurations 3 and 4 result in significantly higher numbers of created transient replicas compared to configurations 1 and 2 (see assumption A14 in Table 5.25).

As expected, *QA5* and *QA2.3* are increased by almost 2,393% and 2,400%, respectively, for configuration 3 averaged over all three strategies. For the averaged configurations, this increase is even slightly higher with average values of 2,415% for *QA5* and 2,421% for *QA2.3*. This clearly confirms the two assumptions *A20* and *A22*. The integration of MFRTR has the strongest impact on PA in configuration 3, i.e., DSF4SP_6_3. The tremendous overall increase is illustrated in Figure 5.36(a), which shows the evolution of transient replicas over time.

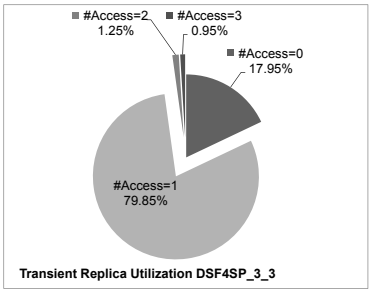
Assumption A24. The use of lazy removal properties of transient replicas moreover results in a decreased pre-replication failure. Compared to DSF4SP_2, DSF4SP_5, and DSF4SP_8, the integration of MFRTR reduces the relative pre-replication failure averaged over all strategy assemblies by a value of 5.4% for configuration 3 and even 9.4% for the averaged configurations. MFRTR has the strongest impact on MPP. Thus, the pre-replication failure of MPP is decreased by values of 8.1% and 10.6% for configuration 3 and the averaged configurations, respectively. In terms of pre-replication failure, PA and CPA worsen MPP by values of 16.4% (absolute failure) and 14.2% (relative failure) for the average case. For configuration 3, this increase is even higher (20.4% and 18.4%). The actual utilisation of transient replicas is illustrated in Figure 5.36(c, d, e).



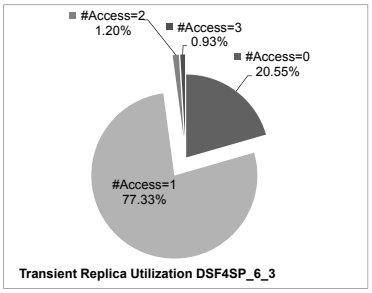
(a)



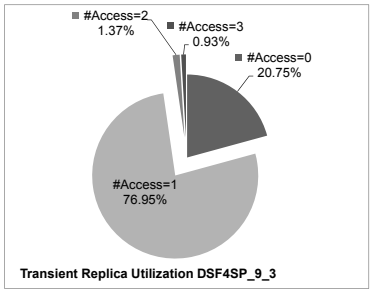
(b)



(c)

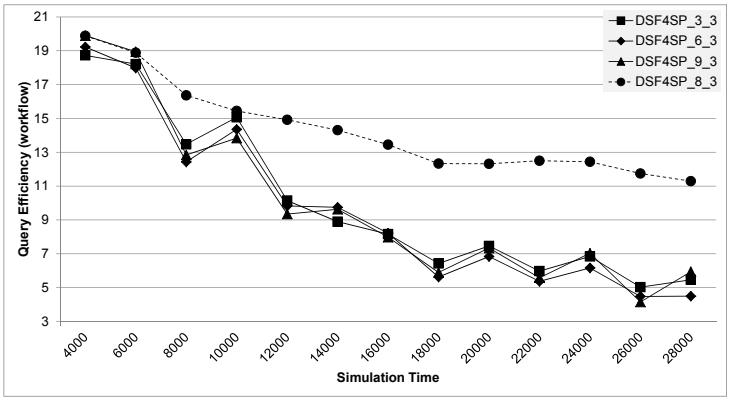


(d)

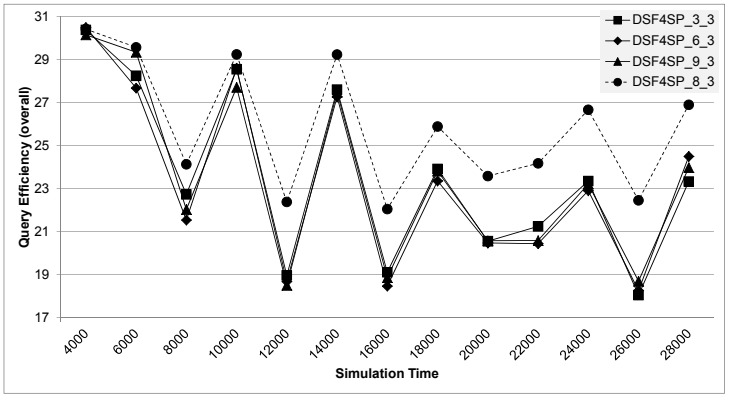


(e)

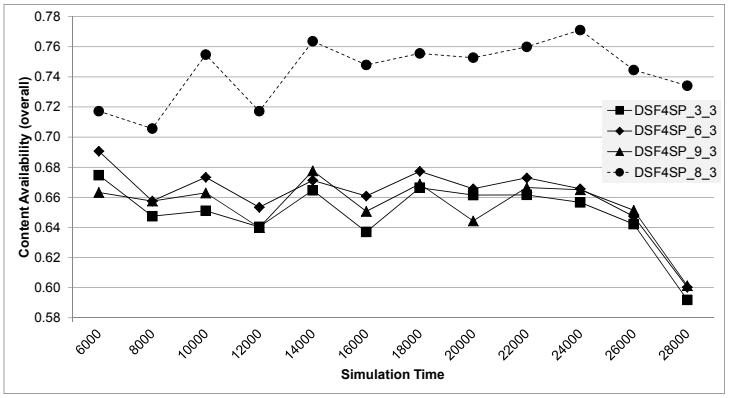
Figure 5.36: Evaluation Results of the Replacement Strategy MFRTR (1/2): (a) transient replica in network, (b) replica in network, (c) transient replica utilisation DSF4SP_3_3, (d) transient replica utilisation DSF4SP_6_3, (e) transient replica utilisation DSF4SP_9_3



(a)



(b)



(c)

Figure 5.37: Evaluation Results of the Replacement Strategy MFRTR 2/2: (a) query efficiency (workflow), (b) query efficiency (overall), (c) content availability (overall)

Assumptions A19, A21, and A23. As stated afore, one would expect a higher number of transient replicas combined with a lower pre-replication failure to result in an increase of both the replica ratio (*QA4*, see *A21* in Table 5.27) and the average number of replicas in the network (*QA2.2*, see *A19* in Table 5.27). However, this is not confirmed by the simulation data. In fact, for MPP and CPA, the attributes *QA4* and *QA2.2* are decreased by almost 11% and 4.9% for configuration 3 and the averaged configurations, respectively, compared to DSF4SP_2 and DSF4SP_8. This appears to be an effect of *content access diversification*, i.e., the higher the number of replicas, the lower the probability that a single replica is accessed often. The contrary increase of DSF4SP_6 compared to DSF4SP_5 most likely results from the relative low amount of data created by PA (see Section 5.4.2.2). Moreover, both for configuration 3 and the averaged configurations, PA outperforms MPP and CPA with respect to *QA4* and *QA2.2*. Hence, with respect to *QA4*, *QA2.2*, and the correlated attribute *QA6*, it can be noted that the integration of cooperation knowledge and the resulting increased complexity lose relevance with increasing transient replica ratios. Figure 5.36(b) depicts the evolution of replicas in the network over time and reveals the reduction of the number of replicas compared to DSF4SP_8_3.

Assumption A26. As expected, query efficiency of workflow-related content requests (*QA8*) is enhanced by the incorporation of MFRTR. The improvement is significant and results in a value averaged over all three strategies of 37.1% for configuration 3 and 30.4% for the average case. Moreover, while the effect appears to be strongest for PA in configuration 3, an analysis of the averaged configurations reveals that CPA achieves the best average results for *QA8* (best value measured for configuration 4). Hence, while the main statement of assumption *A26* is reflected by the simulation data, the assumption of MPP benefiting most from the integration of MFRTR is not confirmed.

The result also applies to *QA11* with MFRTR leading to an average improvement of 24.2% for configuration 3 and 15.6% for the averaged configurations compared to the strategy assemblies 2, 5, and 8. Consequently, the enhancement of query efficiency nicely balances the increasing transient replica ratios. This relation is also reflected in Figure 5.37(a). While the results for strategy assemblies 3, 6, 9, and 8 are rather close for the first 6,000 simulation steps, there is a considerable gap from simulation step 10,000 onwards.

Assumption A25. Also, the overall query efficiency (*QA7*) is improved by MFRTR. Averaged over all three strategies, the actual improvement rate approximates 10% for both configuration 3 and the average case compared to the strategy assemblies 2, 5, and 8. Looking at the averaged configurations, CPA benefits most from the activation of lazy removal properties of transient replicas. The evolution of *QA7* over time is depicted in Figure 5.37(b) and reveals that while MFRTR slightly improves the overall query efficiency, the curve shapes with MFRTR still very much reflect the pattern explained in Section 5.4.2.2. Interestingly, the impact of MFRTR on *QA7* is below one third of the impact on *QA8*. This might be caused by the replication strategies hoarding (transient) replicas close to nodes executing workflows and not optimising non-workflow-related content ac-

cess by any direct means. Even further, the relative overall query efficiency (*QA10*) has worsened by up to 10.2% compared to DSF4SP_2, DSF4SP_5, and DSF4SP_8. Hence, unlike *QA8* and *QA11*, the reduced enhancement of *QA7* does not balance the increased number of transient replicas.

Assumption A27. Finally, the integration of MFRTR has a considerable impact on both absolute and relative content availability. However, as opposed to the expected behaviour, content availability decreases by approximately 11% on average (see also Figure 5.37(c)). This effect is even stronger for *QA15*, which takes into account replica and transient replica ratios and results in a decrease of almost 23%. This unexpected behaviour can be explained as follows. First, as stated above, the replication strategies hoard (transient) replicas around nodes capable of executing workflows and do not aim at an equal distribution of (transient) replicas across all nodes. Moreover, while the routing concept of the P2P overlay network Pastry relies on similarity of content and node identifiers (see Section 5.3.3), MFRTR solely takes into consideration request rates. Hence, identifier similarity is taken into account indirectly, only. While this does improve query efficiency in particular of direct requests, it might have a negative impact on content availability. Even further, the replacement mechanism is non-recursive (see Section 5.3.4.6). Hence, an increased number of (transient) replicas might result in less successful content replacements and / or content replacements to less suitable nodes (in terms of identifier similarity). The effect of replacement operations taking into consideration identifier similarity as well as of recursive replacement concepts should be addressed by future research.

Conclusion

Comparing the three strategies, one can again observe a positive balancing of increased transient replica ratio and enhancement of query efficiency. PA and CPA enhance overall query efficiency as well as query efficiency of workflow-related content request compared to MPP concerning both absolute and relative quality attributes. For all attributes, PA and CPA still lead to very close results with an average variation below 1%. Furthermore, the difference between MPP and PA / CPA in terms of the total number of created transient replicas as well as the transient replica ratio decreased with the activation of lazy removal properties of transient replicas.

Interestingly, while CPA most benefits from the incorporation of MFRTR in terms of query efficiency, PA achieves the highest transient replica utilisation. Even though PA and CPA show quasi equal pre-replication failure rates, PA increases the replica ratio as well as the average number of replicas in the network by approximately 2.5%. Hence, with regard to transient replica utilisation, the cooperative approach of CPA loses relevance with increasing transient replica ratios.

Overall Result
The incorporation of MFRTR results in a significant increase of the transient replica ratio and a reduction of the pre-replication failure. This positively affects query efficiency of workflow-related content requests, which even balances the increasing transient replica ratio. While overall query efficiency is improved as well, the focus of the workflow-based replication strategies leads to a decrease in relative overall query efficiency. In terms of content availability, the incorporation of MFRTR yields worse results compared to MFR. In conclusion, following the results of the analysis presented in Section 5.4.2.2, PA may be rated as the strategy of choice. This is reasoned by the utilisation of transient replicas achieved by PA as well as the similarity of PA and CPA in terms of query efficiency given the communication overhead of CPA.

5.4.2.4 Content-Class-based Content Placement

The third group of content placement strategies consists of all approaches that take into consideration content classes. This includes DSF4SP_4, DSF4SP_7, and DSF4SP_10 that extend MPP, PA, and CPA, respectively, with the strategies CC and ECR. Also, DSF4SP_1 is captured to exclusively assess the effect of the content-class-based placement strategies. For reasons of comparability, configuration 3 is used as primary configuration option in the following analysis.

The modelled distribution of initiator- and requester-assigned content classes is presented in Section 5.2.5 (see Table 5.16). Thus, while workflow-related content requests resulting from activity processing and pre-replication are classified as *CC_QUERY_EFFICIENCY*, all other content class assignments follow the distribution 20%, 50%, and 30% for content class *CC_DEFAULT*, *CC_AVAILABILITY*, and *CC_QUERY EFFICIENCY*, respectively.

Expected Effect of the Configuration Options

As described in Section 5.4.1.2, the configurations 1 and 3 set the parameters Ω, Ψ, and Φ of CC to half of the values assigned by the configurations 2 and 4, and double the value of the threshold parameter Γ used for controlling reactive replication. Hence, it can be assumed that the configurations 2 and 4 approximately double the ratios of replicas and transient replicas compared to the configurations 1 and 3.

Expected Results

Properties of Replicas and Transient Replicas. The distribution of content classes can be used for comparing the potential number of transient replicas created by workflow- and non-workflow-driven replication strategies. On the one hand, the pure workflow-based replication strategies are supposed to create a single transient replica for each workflow-related content need at the maximum. Given the calculation presented in Section 5.2.5, this leads to a maximum value of 102,418 transient replicas. Yet, this calculation abstracts from failures during workflow processing, unnecessary pre-replication, and the availability of activity-related content needs. For reasons of simplicity, these aspects are taken into consideration by a 25% reduction of the maximum value resulting in a total number of 76,813 transient replicas. Moreover, according to the results presented in Section 5.4.2.3, the incorporation of lazy removal properties of transient replicas leads to a reduction of the total number of created transient replicas by approximately 50%. This yields a total number of 38,407 transient replicas.

On the other hand, DSF4SP_1 creates 2 (3) transient replicas in configuration 1 and 3 (2 and 4) for each *PUT* operation classified as *CC_QUERY_EFFICIENCY*. Given that 30% of the total number of 102,728 *PUT* events[63] are classified as *CC_QUERY_EFFICIENCY*, this results in an approximate number of 61,636 (92,455) transient replicas.

Consequently, DSF4SP_1 is expected to increase the total number of created transient replicas compared to workflow-based replication strategies combined with MFRTR. Moreover, the strategy assemblies 4, 7, and 10 are expected to strongly increase the total number of created transient replicas, because of the combination of workflow- and content-class-based replication strategies. Both effects are assumed to apply to the transient replica ratios (*QA5*) as well as the average number of transient replicas in the network (*QA2.3*).

The increasing number of transient replicas is expected to decrease the utilisation of the latter in terms of the absolute pre-replication failure (*QA6*). Also, the relative failure is supposed to increase, because of CC creating transient replicas without a short-term request prediction and the simulation model not covering access patterns that completely follow content class assignments. Even further, for reasons of comparability, the TTL interval configuration is optimised towards workflow execution and not adapted to non-workflow-related access properties such as inter-access times.

In contrast to the purely workflow-driven replication strategies, CC provides means for explicit creation of replicas. For this reason, the replica ratio (*QA4*) as well as the average number of replicas in the network (*QA2.2*) is expected to increase significantly. Furthermore, as opposed to *QA5* and *QA2.3*, the difference between strategy assemblies 1 and 4, 7, 10 is expected to be minor because of the low number of replicas generated by the workflow-based replication strategies.

[63] Note that this is the actual number of *PUT* events measured for each simulation run, which slightly varies from the expected number of *PUT* events stated in Section 5.2.5.

Query Efficiency. In terms of query efficiency of workflow-related content requests, the incorporation of content-class-based placement strategies is expected to slightly improve *QA8* over strategy assemblies 3, 6, and 9. This is reasoned by the higher number of (transient) replicas and the increased probability of workflow-related content needs being stored on-board of nodes triggering workflow execution. Hence, the impact should be highest for MPP, which limits pre-replication of content needs to a single workflow branch. DSF4SP_1 is assumed to worsen *QA8*, because it does not provide any means for active optimisation of workflow-related content requests.

The overall query efficiency *QA7* is expected to improve, because of the higher (transient) replication degrees. With respect to DSF4SP_1, the effect depends on the relation between the enhancement of non-workflow-related content requests and the reduction of query efficiency of workflow-related requests (remember that DSF4SP_1 does not address the latter).

The quality attribute *QA9*, which takes into account content requests with assigned content class *CC_QUERY_EFFICIENCY*, only, is assumed to improve *QA7*, because of the workflow-related content requests playing a more dominant role. Similarly, *QA9* is expected to not achieve the results of *QA8* that solely captures workflow-related content requests. In any case, ECR trying to enhance locality of content classified as *CC_QUERY_EFFICIENCY* is supposed to positively affect query efficiency.

Overall, the aforementioned effects are expected to decrease for the relative quality attributes *QA10*, *QA11* and *QA12*, because of the growth of the number of (transient) replicas.

Content Availability. In general, it can be expected that the combination CC, ECR enhances content availability *QA13* by placing replicas in the leaf set of primary providers given node churn and considering content classes when determining replacement candidates. Moreover, taking into account the results presented in Section 5.4.2.3, one can assume strategy assembly 1 to achieve better content availability than assemblies 4, 7, 10, because of the lower number of transient replicas generated by the former.

Concerning content-class-based content availability (*QA14*), a general assumption can hardly be made. On the one hand, *QA14* may benefit from an increased number of replicas in case of repetitive requests. On the other hand, *QA14* does not take into account workflow-related content requests.

With respect to the relative properties *QA15* and *QA16*, the impact of the content-class-based placement strategies is supposed to reduce considerably due to the high replica ratios effected by CC. The expected results are summarised in Table 5.29.

Simulation Results

The simulation data of the strategy assemblies DSF4SP_1_3, DSF4SP_4_3, DSF4SP_7_3, and DSF4SP_10_3 are summarised in Table 5.30 and illustrated by Figures 5.38 and

ID	Quality Attribute	Assumption
A28	QA2.2	CC leads to a higher number of replicas in the network (similar values for strategy assemblies DSF4SP_{1, 4, 7, 10}).
A29	QA2.3	DSF4SP_1 leads to a higher number of transient replicas in the network than DSF4SP_{3, 6, 9}.
A30	QA2.3	DSF4SP_{4, 7, 10} lead to a higher number of transient replicas in the network than DSF4SP_{1, 3, 6, 9}.
A31	QA4	CC leads to a higher replica ratio (similar values for strategy assemblies DSF4SP_{1, 4, 7, 10}).
A32	QA5	DSF4SP_1 leads to a higher transient replica ratio than DSF4SP_{3, 6, 9}.
A33	QA5	DSF4SP_{4, 7, 10} lead to a higher transient replica ratio than DSF4SP_{1, 3, 6, 9}.
A34	QA6	CC leads to a reduced transient replica utilisation.
A35	QA7	DSF4SP_{4, 7, 10} leads to an improved overall query efficiency compared to DSF4SP_{3, 6, 9}.
A36	QA8	DSF4SP_{4, 7, 10} leads to an improved workflow-related query efficiency (strongest for MPP) compared to DSF4SP_{3, 6, 9}.
A37	QA8	DSF4SP_1 leads to a reduced workflow-related query efficiency compared to DSF4SP_{3, 6, 9}.
A38	QA13	CC / ECR leads to an improved content availability compared to DSF4SP_{3, 6, 9}.
A39	QA13	DSF4SP_1 leads to a higher content availability than DSF4SP_{4, 7, 10}.

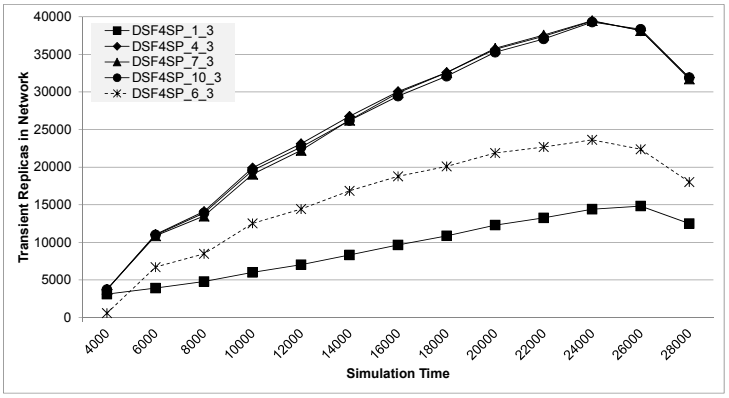
Table 5.29: Simulation Analysis: Expected Results

	QA2.2	QA2.3	QA4	QA5	QA6 (0)	QA7	QA8	QA9	QA10	QA11	QA12	QA13	QA14	QA15	QA16
DSF4SP_1_3	20,988.21	8,813.18	66.23%	27.81%	97.94%	28.18	25.23	27.03	34.40	31.26	32.97	70.34%	72.09%	59.25%	57.82%
DSF4SP_1_4	34,771.93	17,793.51	105.56%	54.02%	95.04%	25.93	20.49	23.58	35.15	27.97	31.57	70.44%	72.38%	52.78%	52.55%
DSF4SP_4_1	21,158.12	19,752.85	66.84%	62.40%	62.49%	23.63	13.94	20.45	34.36	21.50	30.12	68.72%	70.68%	50.84%	50.66%
DSF4SP_4_2	35,013.66	27,013.02	105.78%	81.61%	69.47%	22.59	12.78	19.18	33.32	19.48	28.11	69.40%	71.02%	48.67%	48.57%
DSF4SP_4_3	21,322.40	24,684.16	67.43%	78.06%	60.12%	21.65	8.26	17.15	31.85	12.87	25.47	69.17%	70.70%	49.99%	49.98%
DSF4SP_4_4	34,788.33	31,592.41	105.14%	95.48%	65.12%	20.84	7.59	16.26	31.26	11.76	24.30	70.62%	71.76%	48.40%	48.46%
DSF4SP_7_1	21,210.25	20,244.75	66.87%	63.83%	63.86%	23.43	13.90	20.31	33.93	21.29	29.79	68.46%	70.16%	50.58%	50.18%
DSF4SP_7_2	34,908.53	28,001.61	106.03%	85.05%	69.72%	22.67	12.81	19.19	33.37	19.45	28.15	69.50%	71.18%	48.71%	48.72%
DSF4SP_7_3	21,210.26	24,460.24	66.57%	76.77%	60.65%	21.62	8.01	16.98	32.04	12.47	25.28	68.70%	70.16%	49.63%	49.59%
DSF4SP_7_4	34,889.15	31,498.92	104.72%	94.54%	65.90%	20.77	7.35	16.14	31.08	11.43	24.04	70.52%	72.22%	48.36%	48.83%
DSF4SP_10_1	21,401.03	20,327.28	67.27%	63.90%	63.10%	23.70	14.05	20.55	34.50	21.67	30.35	69.47%	71.26%	51.02%	51.02%
DSF4SP_10_2	34,842.94	27,080.87	105.13%	81.71%	69.20%	22.84	12.79	19.31	33.68	19.58	28.29	69.87%	71.39%	49.06%	48.98%
DSF4SP_10_3	21,277.54	24,434.51	67.13%	77.09%	60.29%	21.11	7.77	16.79	31.02	12.00	24.73	69.50%	70.58%	49.96%	49.60%
DSF4SP_10_4	35,026.21	31,670.02	105.91%	95.77%	65.97%	21.09	7.50	16.35	31.79	11.69	24.50	70.37%	71.68%	48.27%	48.36%

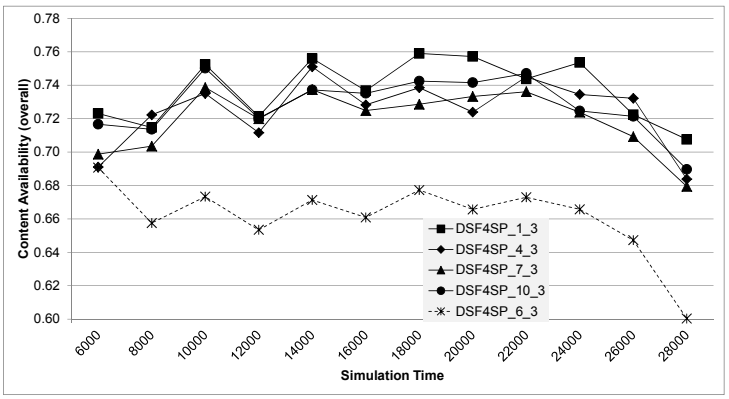
Table 5.30: Evaluation Results of the Content Placement Strategies CC and ECR

5.39. For reasons of comparability, these figures also include the best result of the strategies DSF4SP_3_3, DSF4SP_6_3, and DSF4SP_9_3 for each quality attribute.

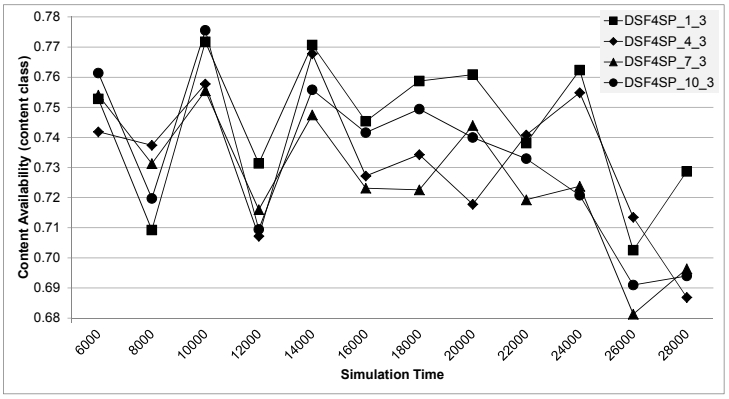
Assumptions A29, A30, A32, and A33. As expected, the content-class-driven strategies increase the total number of created transient replicas. While the strategy assembly 1 increases assembly 6 by 15.7% for configuration 3, this growth reaches a value of 81.5% for the averaged configurations. This is in fact reasonable, since strategy assembly 6 makes use of lazy removal properties of transient replicas. Compared with assembly 5, i.e., pure PA, the total number of transient replicas is decreased by 18.9% for the averaged configurations. Furthermore, as supposed, the assemblies 4, 7, and 10 lead to a considerable average increase of the number of created transient replicas with values of 130.5% for configuration 3 and even 174.5% for the averaged configurations compared to assemblies 3, 6, and 9.



(a)

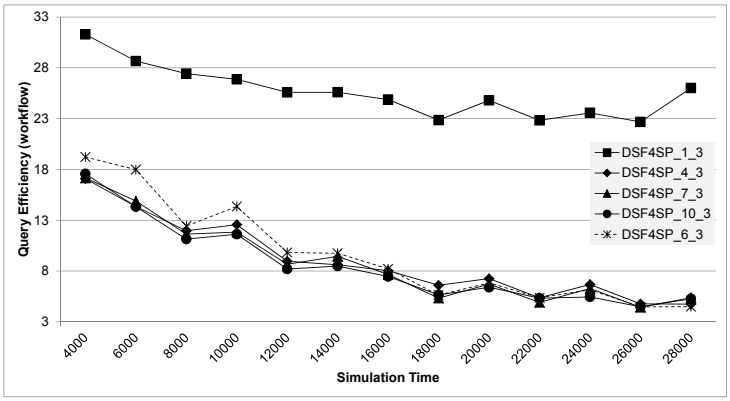


(b)

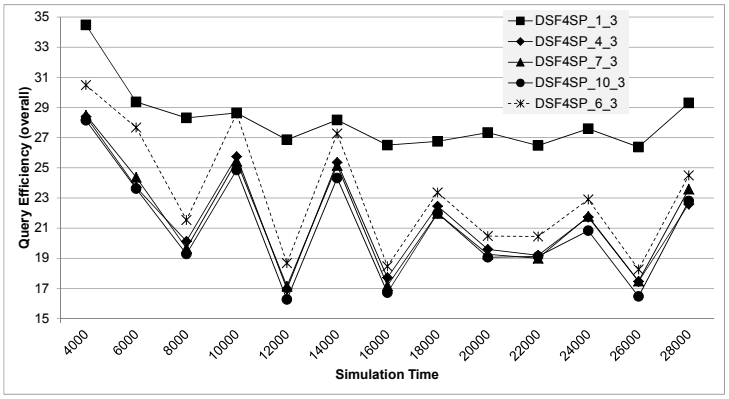


(c)

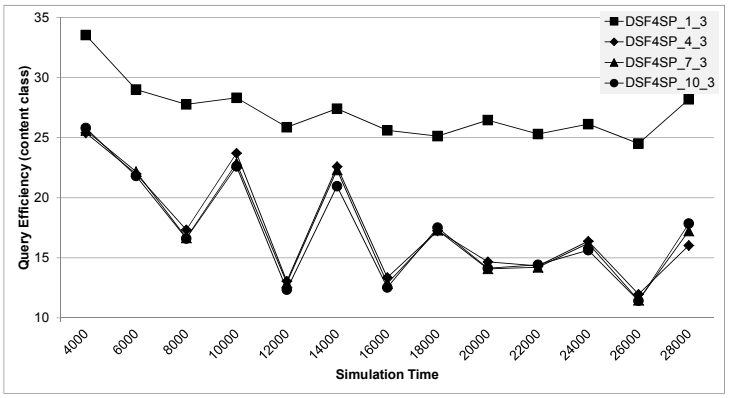
Figure 5.38: Evaluation Results of the Content Placement Strategies ECR and CC (1/2): (a) transient replica in network, (b) content availability (overall), (c) content availability (content class)



(a)



(b)



(c)

Figure 5.39: Evaluation Results of the Content Placement Strategies ECR and CC 2/2: (a) query efficiency (workflow), (b) query efficiency (overall), (c) query efficiency (content class)

This effect also applies to the quality attributes *QA5* and *QA2.3* with an average increase of approximately 69% (configuration 3) and 100% (averaged configurations) for the assemblies 4, 7, and 10 compared to the assemblies 3, 6, and 9. Also, compared to assembly 1, the assemblies 4, 7 and 10 (average configuration) increase the two attributes by an average value of 94.7%. Hence, the assumptions *A30* and *A33* are backed up by the simulation data. Interestingly, for DSF4SP_1, despite the increase of the total number of created transient replicas, *QA5* and *QA2.3* are decreased by almost 40% for configuration 3 compared to DSF4SP_6_3. For the averaged configurations, the results are quite similar (variation below 4.5%).

Assumption A34. This is reasoned by a strong decrease of the transient replica utilisation. Thus, DSF4SP_1 increases the relative pre-replication failure of DSF4SP_6 by 376.7% for configuration 3 and almost 500% for the averaged configurations. While the strategy assemblies 4, 7, and 10 result in very close pre-replication failures, they also considerably increase the failure measured for assemblies 3, 6, and 9 by up to 235% for configuration 3 and up to 354% for the averaged configurations (MPP in both cases). Hence, the simulation data reflect assumption *A34*. An overview of the evolution of the number of transient replicas over time is presented in Figure 5.38(a). It illustrates the strong increase caused by the strategy assemblies 4, 7, 10, as well as the relation between DSF4SP_1_3 and DSF4SP_6_3 explained above.

Assumptions A28 and A31. The expected tremendous increase of the attributes *QA4* and *QA2.2* is also reflected by the simulation data. Thus, the average increase across all four strategies compared to the strategy assemblies 3, 6, 9 approximates values of 15,117% (*QA4*) and 15,350% (*QA2.2*) for configuration 3 as well as 19,924% (*QA4*) and 20,837% (*QA2.2*) for the averaged configurations. While this increase is indeed considerable, it has to be noted that the replica ratio (*QA4*) approximates values of 105% and 67% for configurations 2, 4, and 1, 3, respectively. Hence, the number of replicas created by CC does not significantly exceed the number of content stored in the network. The evolution of replicas over time is very close to the curve shapes of the strategies 4, 7, 10 depicted in 5.38(a). Due to the minor impact of replicas resulting from the workflow-based replication strategies, all four curves are so close that a dedicated chart would not provide any insights.

Assumption A36 and A37. As supposed, query efficiency of workflow-related content requests is further enhanced by the content-class-based strategies. Hence, for the strategy assemblies 4, 7, 10 in configuration 3 and the averaged configurations, an average improvement of 9.7% and 11.7%, respectively, is measured compared to the strategy assemblies 3, 6, 9. According to the results of the averaged configurations, it is MPP (as expected) that benefits most from the approach. All three assemblies achieve very close results with less than 1.2% variance in the averaged configurations case. Also, the assumed worsening of DSF4SP_1 is reflected by the simulation data (almost 92% for the averaged configurations compared to DSF4SP_6).

This also applies to *QA11*, which reveals a worsening of 153.3% (configuration 3) and 75.6% (averaged configurations) for DSF4SP_1 compared to DSF4SP_6. Looking at the results of the averaged configurations, the three strategy assemblies 4, 7, 10 enhance *QA11* compared to the assemblies 3, 6, 9 by 4.5% on average. Consequently, these strategies are able to effectively trade off the increasing number of (transient) replicas with the enhancement of query efficiency of workflow-related requests.

Yet, it needs to be noted that CC leads to a rather uniform distribution of replication degrees per content. Combined with the natural logarithm used in the calculation of relative query efficiency attributes, this results in a reduced impact of the increased number of (transient) replicas and likely affects the positive results measured for *QA11*.

Assumption A35. The simulation data also reflect the expected impact on *QA7* and *QA10*. For the strategy assemblies 4, 7, 10, an average enhancement of *QA7* by 9.7% is measured compared to the strategy assemblies 3, 6, 9 (averaged configurations). Interestingly, the effect of the content-class-driven placement strategies appears to be lower for *QA7* than for *QA8* even though all requests are addressed. Yet, with regard to the absolute enhancement, the strategy assemblies 4, 7, 10 improve the strategy assemblies 3, 6, 9 by 1.4 simulation steps for *QA8* and almost 2.4 simulation steps for *QA7* (averaged configurations).

The impact on *QA10* resembles the above-presented effect on *QA11*. While the effect is considerably reduced compared to *QA7*, the three strategies are capable of enhancing overall relative query efficiency compared to the strategy assemblies 3, 6, and 9. Again, this confirms a successful balancing of the number of (transient) replicas and the enhancement of *QA7*.

Furthermore, for the overall query efficiency, the divergence of DSF4SP_1 is reduced considerably. Concerning the averaged configurations, the latter worsens DSF4SP_6 in terms of *QA7* and *QA11* by only 10% and 4.9%, respectively.

The effect of the content-class-based strategies on *QA8* and *QA7* is illustrated in Figures 5.39(a) and 5.39(b), respectively. The significant gap between strategy assembly 1 and the assemblies 4, 7, 10, 6 measured for *QA8* is clearly reduced for *QA7*. Also, for the latter, one can again see the pattern recognised and explained for Figures 5.35(b) and 5.37(b). Moreover, the results measured for *QA9* are in between *QA8* and *QA7*. As depicted in Figure 5.39(c), the curve shapes very much reflect the patterns seen in Figure 5.39(b) for the overall query efficiency, but are slightly below, i.e., better, in all cases. Yet, in particular for strategy assemblies 4, 7, 10, the measured values are considerably above, i.e., worse, the results measured for *QA8*. With respect to the averaged configurations, the three strategy assemblies 4, 7, 10 show an average variation below 0.6% for *QA9* and 0.7% for *QA12*. Hence, while the simulation data reflect the expected behaviour for *QA9* and *QA12*, the two quality attributes provide no meaningful insights.

Assumptions A38 and A39. In terms of content availability, DSF4SP_1 achieves the highest results, which are slightly reduced by the strategy assemblies 4, 7, 10 by an

average value of 1.2%. Moreover, the latter improve the strategies presented in Section 5.4.2.3 by 11% on average (both times averaged configurations). Hence, the values are very close to the results achieved by the pure workflow-based replication strategies with a variance of 2%, only. This demonstrates that (i) MPP, PA, and CPA positively (indirectly) affect content availability given the high portion of workflow-related content requests and (ii) the simulation model does not provide access patterns that fully comply with content class assignments.

The strategy assemblies 4, 7, 10 decrease relative content availability achieved by DSF4SP_1 by 11.7% on average (averaged configurations, *QA15*). Even further, the strategy assemblies 4, 7, 10 decrease *QA15* by an average value of 8.4% for the averaged configurations compared to the assemblies 3, 6, and 9. Hence, the increase of content availability is not well-balanced with the growth of the number of (transient) replicas.

While the relations between the four strategies for *QA13* and *QA15* are reflected by the quality attributes *QA14* and *QA16*, *QA14* shows a minor improvement. Hence, placement of replicas in the repository set of primary providers achieves enhanced content availability. Moreover, in contrast to *QA13*, *QA14* excludes not only content of class *CC_QUERY_EFFICIENCY* but also content classified as *CC_DEFAULT*. Given the configuration of 20% of *PUT* operations being assigned content class *CC_DEFAULT* and the latter not leading to any replication, the difference between *QA13* and *QA14* becomes reasonable. Consequently, one should think of adding a minimum replication degree for content of class *CC_DEFAULT* to compensate the above-explained loss of content availability.

The results are illustrated by Figures 5.38(b) and 5.38(c). The drop of content availability seen at simulation steps 8,000, 12,000, and 16,000 is assumed to result from (i) temporary sub-optimal content placement caused by transient replicas created during workflow execution (see Figure 5.34(a)) and – in particular – (ii) the evolution of nodes in the network presented in Figure 5.32(a). This becomes even more obvious when comparing the pattern with the curve shape of *QA14* for the strategy NO REPLICATION presented in the next section.

Conclusion

The difference of the three workflow-based replication strategies in terms of transient replica ratio and utilisation, as well as overall and workflow-related query efficiency are strongly reduced by the introduction of the content-class-based strategies. The variation of the quality attributes *QA6*, *QA2.2*, and *QA4* is below 1% on average for all three strategies 4, 7, 10. Moreover, while the latter are capable of improving query efficiency considerably over strategy assemblies 3, 6, 9, the impact of MPP, PA, and CPA is decreased with a maximum difference of 1.5% for the averaged configurations. This also applies to content availability with an average variation of 0.6% for *QA13* and *QA15* (averaged configurations).

The selection of any of the strategy assemblies depends on the primary objective of the distributed storage system used in the target scenario. Obviously, the exclusion of the workflow-based replication strategies cannot be recommended because of the strong impact on workflow-related query efficiency (see results of DSF4SP_1 for *QA8* and *QA10*).

Overall Result
The incorporation of the content-class-based strategies leads to higher (transient) replica ratios as well as a lower utilisation of transient replicas. The decreased utilisation of transient replicas is driven by CC creating transient replicas without a short-term request prediction and the simulation model not covering access patterns that completely follow content class assignments. However, the strategy assemblies 4, 7, 10 achieve the best results for overall and workflow-related query efficiency. Also, CC and ECR are able to compensate the negative effect on content availability resulting from the use of MFRTR. Overall, even though PA slightly increases query efficiency over CPA, the latter achieves marginally better results for the defined content availability attributes. Given the general low level of content availability, DSF4SP_10 may be rated as the strategy of choice.

5.4.2.5 Comparison with Related Work

This section analyses the simulation data of related work and provides a comparison of the latter with the proposed content placement strategies. According to the simulation results of each of the three groups, this includes CPA for group 1 (DSF4SP_8, see Section 5.4.2.2), PA for group 2 (DSF4SP_6, see Section 5.4.2.3), and CPA for group 3 (DSF4SP_10, see Section 5.4.2.4). Prior to the actual comparison, the different configurations of the strategies MDCDN and CACHING (note that NO REPLICATION is simulated in a single configuration, only) presented in Section 5.4.1.3 are discussed. Given the defined quality attributes, the most suitable configuration of each strategy is selected and compared with the above-listed strategy assemblies of the proposed content placement strategies. The simulation data of the related work MDCDN, CACHING, and NO REPLICATION are summarised in Table 5.31.

Effect of the Configuration Options

MDCDN. As discussed in Section 5.4.1.3, MDCDN is simulated in a rather risk-averse (MDCDN_2) and a more optimistic configuration (MDCDN_1). Driven by the more frequent determination of periodic indices as well as the shorter reconfiguration interval, MDCDN_1 almost doubles the replica ratio (*QA4*) as well as the average number of replicas in the network (*QA2.2*) compared to MDCDN_2. This increase positively affects both

	QA2.2	QA2.3	QA4	QA5	QA7	QA8	QA10	QA11	QA13	QA15
MDCDN_1	1,773.47		5.68%		30.12	26.76	33.55	30.64	70.81%	69.01%
MDCDN_2	888.46		2.86%		30.97	27.39	33.72	30.34	71.55%	72.56%
CACHING_1	6,282.76		20.11%		27.35	20.38	31.56	24.02	68.99%	61.37%
CACHING_2	1,087.33		3.49%		30.03	25.71	32.61	28.16	70.66%	70.36%
CACHING_3	27,281.61		83.15%		23.41	13.01	29.38	16.85	67.94%	51.16%
NOREP	0.00		0.00%		30.89	28.75	30.89	28.75	75.75%	82.86%
DSF4SP_8_avg	141.29	504.07	0.45%	1.62%	27.08	17.09	30.30	20.10	70.49%	69.82%
DSF4SP_6_avg	136.64	12,735.99	0.44%	40.96%	24.59	11.92	33.14	16.87	62.48%	53.94%
DSF4SP_10_avg	28,136.93	25,878.17	86.36%	79.62%	22.18	10.53	32.75	16.23	69.80%	49.58%

Table 5.31: Evaluation Results of Related Work

QA8 and *QA7*, which are enhanced by values of 2.3% and 2.8%, respectively. In terms of relative query efficiency, MDCDN_1 worsens MDCDN_2 by 1% concerning *QA11*, while it improves the latter by 0.5% concerning *QA10*. This results from the lower improvement of *QA8* compared to *QA7* achieved by MDCDN_1. Finally, MDCDN_2 increases (relative) content availability by (5.1%) 1% compared to MDCDN_1.

The relation of replica ratio and content availability measured for MDCDN is in-line with the evaluation results presented before. Hence, given that the replica ratio of MDCDN_1 is still below 6%, the configuration MDCDN_1 is used as representative for comparing MDCDN with the proposed content placement strategies.

CACHING. The strategy CACHING is simulated in three configurations with threshold values of 2 (configuration 1), 4 (configuration 2), and 0 (configuration 3, pure caching as defined in Section 2.1.2). As expected, the lower the threshold for creating replicas, the higher the number of both *QA4* and *QA2.2*. Thus, compared to CACHING_1, CACHING_2 reduces *QA4* and *QA2.2* by almost 82%, while CACHING_3 increases both attributes by an average value of 324%. Again, replica ratio directly impacts query efficiency. In comparison with CACHING_1, CACHING_2 worsens *QA8* and *QA11* by 26.1% and 17.2%, respectively. In contrast, CACHING_3 improves CACHING_1 by values of 36.2% (*QA8*) and 29.9% (*QA11*). In terms of overall (relative) query efficiency, this effect can also be seen but with a reduced variation, which is reasoned by the repetitive execution of workflows as defined in the simulation model. Finally, concerning content availability, one can again observe the negative impact of high replica ratios discussed in Section 5.4.2.3. Hence, on the one hand, CACHING_2 improves CACHING_1 by 2.4% and even 14.7% with regard to *QA13* and *QA15*, respectively. CACHING_3, on the other hand, further worsens CACHING_1 by 1.5% (*QA13*) and 16.6% (*QA15*).

Taking into account the strong focus on query efficiency of the proposed content placement strategies, the improvement of CACHING_3 in terms of query efficiency outweighs its loss of content availability. For this reason, CACHING_3 used for comparing CACHING with the proposed content placement strategies.

The results of MDCDN_1, CACHING_3, and NO REPLICATION, as well as the selected content placement strategies DSF4SP_8, DSF4SP_6, and DSF4SP_10 are depicted in Figures 5.40 and 5.41. While the below analysis makes use of the averaged configurations of the three selected content placement strategies, configuration 3 is again used in the illustrations for reasons of comparability.

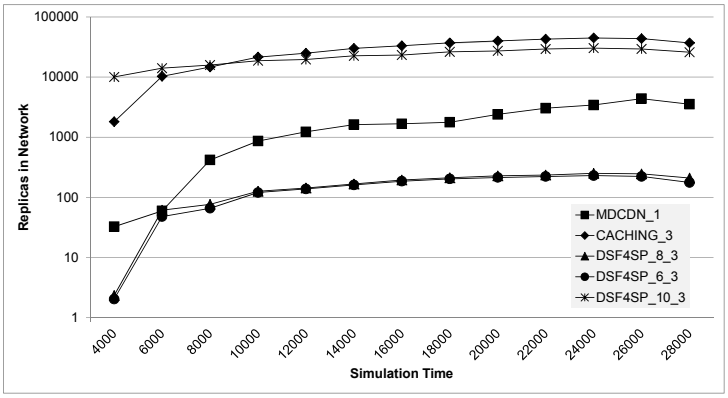
Simulation Results

Properties of Replicas and Transient Replicas. Given that the persisting of transient replicas represents the only means of DSF4SP_6 and DSF4SP_8 for creating replicas, it is obvious that their results of both *QA4* and *QA2.2* are considerably below all other strategies except NO REPLICATION. Thus, DSF4SP_6 and DSF4SP_8 reduce the attributes *QA4* and *QA2.2* by an average value of 92% compared to MDCDN_1 and even 99.5% compared to CACHING_3. In contrast, the content-class-driven strategies applied by DSF4SP_10 approximate the results measured for CACHING_3 in terms of *QA4* and *QA2.2* with an average variation of 3.5%. Compared to MDCDN_1, DSF4SP_10 increases both attributes by approximately 1,454%.

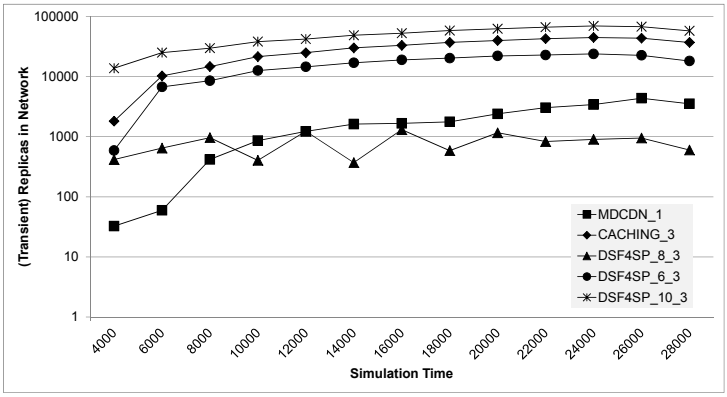
Taking into consideration the average number of transient replicas in the network (*QA2.3*), the difference in the average number of (transient) replicas changes considerably. While DSF4SP_8 decreases MDCDN_1 by 64%, the latter is increased by DSF4SP_6 by almost 625% driven by the incorporation of lazy removal properties of transient replicas. Even further, DSF4SP_10 increases MDCDN_1 by 2,945%. Comparing the proposed strategies with CACHING_3, the difference results in a decrease of 97% for DSF4SP_8 and 53% for DSF4SP_6, as well as an increase of 98% for DSF4SP_10.

Hence, concerning the overall number of (transient) replicas stored by the different strategies in the network, DSF4SP_8 achieves the by far lowest value followed by MDCDN_1. The remaining approaches are sorted in the order DSF4SP_6, CACHING_3, DSF4SP_10, with the latter achieving the by far highest value. The relation between the different approaches taking into consideration replicas only as well as both replicas and transient replicas are presented in Figures 5.40(a) and 5.40(b), respectively.

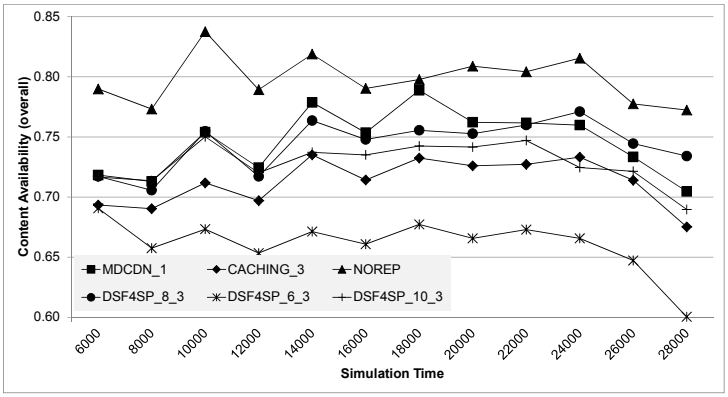
Workflow-Related Query Efficiency. In terms of workflow-related query efficiency, the proposed strategies achieve an average improvement of 50.7% and 54.2% over MDCDN_1 and NO REPLICATION, respectively. For *QA11*, the results are quite similar with values of approximately 42.1% and 38.3%. Of course, the differences between the three strategy assembly groups represented by DSF4SP_6, DSF4SP_8, and DSF4SP_10 explained in the above sections are still valid. With regard to CACHING_3, the results vary considerably. While DSF4SP_6 and DSF4SP_10 still improve CACHING_3 in terms of *QA8* by 8.3% and 19.1%, respectively, DSF4SP_8 worsens the latter by more that 31%. Concerning *QA11*, DSF4SP_8 worsens CACHING_3 by 19.3% (reduction caused by the lower number of (transient) replicas), whereas DSF4SP_10 achieves a minor improvement of 3.6%. The results of CACHING_3 and DSF4SP_6 are almost identical.



(a)

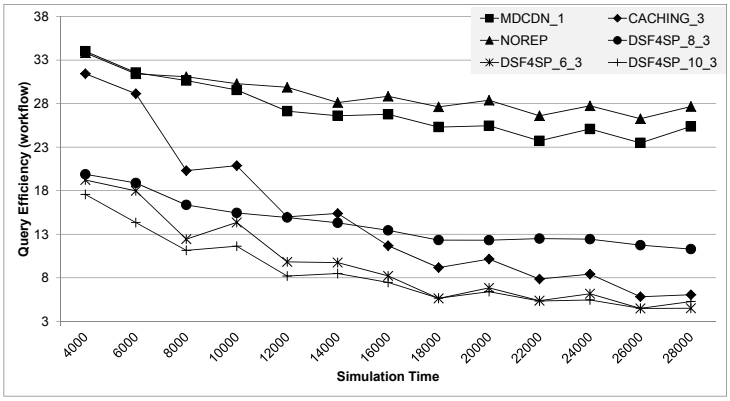


(b)

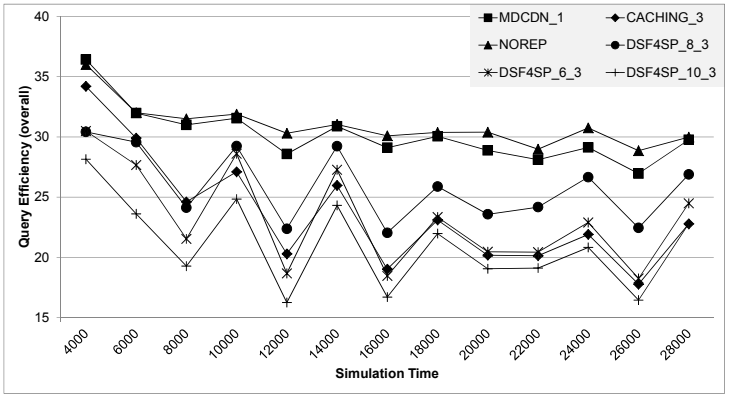


(c)

Figure 5.40: Evaluation Results of Related Work (1/2): (a) replica in network, (b) replica and transient replica in network, (c) content availability (overall)



(a)



(b)

Figure 5.41: Evaluation Results of Related Work (2/2): (a) query efficiency (workflow), (b) query efficiency (overall)

The integration of lazy removal properties of transient replicas pays back, whereas the low replication degree of DSF4SP_8 shows significant drawback. Of course, DSF4SP_10 achieves the best results driven by its strong replication approach. However, its effect is relativised by *QA11* even though the latter weights the number of (transient) replicas by means of the natural logarithm, only. The evolution of query efficiency of workflow-related content requests is illustrated in Figure 5.41(a). Most interestingly is the curve shape of CACHING_3. It falls significantly in the first 12,000 simulation steps and even comes below the curve of DSF4SP_8 after simulation step 14,000. Yet, even though CACHING_3 consistently approaches the curves of DSF4SP_6 and DSF4SP_10 it does not come below the latter. Finally, the comparison between MDCDN_1 and the baseline approach NO REPLICATION reveals the minor impact of MDCDN_1 on workflow-related query efficiency.

Overall Query Efficiency. In terms of overall query efficiency (*QA7* and *QA10*), one can again observe an enhancement of the proposed strategies compared to MDCDN_1. While the average improvement approximates a value of 18% for *QA7*, *QA10* is enhanced by an average value of 4.4%, only. In particular, the increased (transient) replica ratios of DSF4SP_6 and DSF4SP_10 result in low improvements of 1.2% and 2.4%, respectively. Even further, while the proposed strategies all improve NO REPLICATION by an average value of 20.3% concerning *QA7*, this effect is not reflected by *QA10*. Again, driven by the (transient) replica ratios, only DSF4SP_8 slightly enhances NO REPLICATION (1.9%), while both DSF4SP_6 and DSF4SP_10 miss the baseline by 7.3% and 6%, respectively.

With regard to CACHING_3, neither DSF4SP_6 nor DSF4SP_8 enhance overall query efficiency. Yet, taking into consideration that both DSF4SP_6 nor DSF4SP_8 focus on workflow-related content requests, while CACHING_3 improves all content requests equally, this result is reasonable. Thus, DSF4SP_6 and DSF4SP_8 worsen CACHING_3 in terms of *QA7* (*QA10*) by 5.1% (12.8%) and 15.7% (3.1%), respectively. Also, while DSF4SP_10 improves *QA7* compared to CACHING_3 by 5.2%, its high (transient) replica ratio leads to a worsening of *QA10* by 11.5%.

The results are illustrated in Figure 5.41(b). Driven by the repetitive execution of workflows, the three proposed strategies as well as CACHING_3 show curve shapes that match the workflow execution pattern (see also Figure 5.34(a)). With a minor effect, this relation can also be observed for MDCDN_1. Moreover, one can see a close similarity between CACHING_3 and DSF4SP_6 in the interval [16,000; 26,000], with CACHING_3 being below the latter for most measurements.

Content Availability. With regard to content availability, the impact of the number of (transient) replicas described in Section 5.4.2.3 is again confirmed. Consequently, its NO REPLICATION that achieves the highest content availability, which is worsened by the proposed strategies by an average value of 10.8%. Compared to MDCDN_1, this effect is reduced. While DSF4SP_6 reduces *QA13* by 11.8%, both DSF4SP_8 and DSF4SP_10 achieve a level of content availability that is very close to the results measured for MDCDN_1 (see also Section 5.4.2.4). In comparison with CACHING_3, DSF4SP_6 reduces *QA13* by 8%, whereas DSF4SP_8 and DSF4SP_10 approximate an average improvement of 3.2%. Hence, the pure caching concept applied by CACHING_3 worsens content availability compared to DSF4SP_10 despite the considerably higher number of (transient) replicas maintained by the latter.

This effect also applies to relative content availability (*QA15*). Since *QA15* is subject in particular to replica ratios, the differences are further increased for strategies that maintain a high number of replicas. MDCDN_1 is worsened by DSF4SP_6 and DSF4SP_10 by 21.8% and 28.2%, respectively. In contrast, the former is slightly improved by DSF4SP_8, because of the close results of the two in terms of *QA7* and the significantly lower replica ratio of DSF4SP_8. This relation is similar for CACHING_3, which achieves low relative content availability because of its high replica ratio.

It can be concluded that replication on the provider side leads to better content availability than pure replication on the requester side applied by CACHING as well as DSF4SP_6 and DSF4SP_8. Moreover, content availability decreases with an increasing number of (transient) replicas placed on the requester side. This not only applies to DSF4SP_6 and DSF4SP_8 (see Section 5.4.2.3), but can also be observed for the three configurations of CACHING. The results are illustrated in Figure 5.40(c).

Conclusion

Overall, the concept of replication degrees following content popularity applied by CACHING yields very positive results in terms of overall query efficiency. However, the effect strongly depends on the setting of the threshold parameter. Looking at the configuration CACHING_1, the outcome is worse the results of the proposed content placement strategies for almost all quality attributes. In contrast, CACHING_3 leads to a very high number of replicas that might not suite the domain of smart products in all cases. Indeed, the proposed strategies create high transient replica ratios. However, as opposed to replicas, transient replicas only "fill up" available storage capacity of smart products and do not result in any additional replacement overhead.

MDCDN is very close to the baseline represented by the assembly NO REPLICATION. This applies to both configurations MDCDN is simulated with. However, because of the resource limitation of smart products, a further shortening of the reconfiguration period of the statistical demand forecasting method applied by MDCDN in order to increase replica ratios does not appear feasible given the complexity of the recursive calculation. Aggregating and centralising the calculation to either specific smart products (e.g., super nodes assuming hierarchic structures) or even to the infrastructure would address the resource issue of smart products. This should be simulated and analysed in future research.

Overall Result
The strategy CACHING_3 achieves very good results in terms of overall and workflow-related query efficiency and even outperforms DSF4SP_8. However, CACHING_3 leads to a high number of replicas maintained in the network. The comparison with DSF4SP_6 points out the value of transient replicas and the replacement strategy MFRTR. The results achieved by DSF4SP_6 are very close to the results of CACHING_3. Yet, DSF4SP_6 leads to a significantly lower average number of replicas maintained in the network. While DSF4SP_10 and CACHING_3 have similar replica ratios, the former improves the latter in almost all query efficiency and content availability attributes. Hence, while DSF4SP_6 and DSF4SP_8 take into consideration resource limitations of smart products without significantly worsening content access compared to CACHING_3, DSF4SP_10 outperforms the

latter in scenarios that enable high replica ratios. Finally, given the complexity of the content access predictor applied by MDCDN, the latter was simulated in a configuration suitable for the typical resources of smart products. Since this configuration does not yield improvements in terms of query efficiency and content availability, optimised deployment approaches should be studied in future research to further analyse the potential of this active strategy.

5.4.3 Conclusion of the Evaluation Study

This section provides a final overview of the evaluation results and gives recommendations on when to apply which of the proposed strategy assemblies. Again, the analysis makes use of the grouping introduced in Section 5.4.2 with each group being represented by the strategy assembly yielding the best results in terms of the defined quality attributes, i.e., DSF4SP_8, DSF4SP_6, and DSF4SP_10. The comparison with related work presented in Section 5.4.2.5 reveals that these assemblies outperform the simulated strategies MDCDN and CACHING for almost all quality attributes.

Workflow-Related Query Efficiency. In terms of workflow-related query efficiency, all of the proposed strategies significantly enhance the baseline given by the assembly NO REPLICATION. DSF4SP_8 leads to an average improvement of 40.6%, which is further extended by DSF4SP_6 and DSF4SP_10 by another 30.3% and 38.4%, respectively. Also, the expected improvement of PA and CPA compared to MPP driven by their more optimistic balancing of transient replica ratios and pre-replication failure is confirmed by the simulation data. While PA and CPA lead to very similar workflow-related query efficiency caused by the adapted setting of the path assessment threshold parameter, CPA shows the expected improvement of transient replica utilisation. Yet, this effect decreases with increasing transient replica ratios. Also, with an overall growth of (transient) replicas, the difference between the three workflow-based strategies shrinks. In any case, the benefit of pre-replicating workflow-related content needs combined with the concept of transient replicas to account for a potential pre-replication failure is confirmed by the conceptual analysis as well as the simulation data.

Overall Query Efficiency. The aforementioned effect, in a reduced way, also applies to overall query efficiency with an improvement of 12.3% for DSF4SP_8, 20.4% for DSF4SP_6, and 28.2% for DSF4SP_10 compared to the assembly NO REPLICATION. Even though neither DSF4SP_8 nor DSF4SP_6 directly addresses overall query efficiency, the positive result is caused by the high portion of workflow-related requests, which constitute 45% of all *GET* requests captured by the simulation model (see Section 5.2.5). DSF4SP_10 directly addresses overall query efficiency. This way, it is even able

to improve the strategy CACHING, which places replicas on the requester side following content popularity.

Content Availability. In contrast, content availability is decreased by all groups compared to NO REPLICATION. DSF4SP_8 leads to a reduction of almost 7%. This can be reasoned by the fact that (i) DSF4SP_8 solely addresses workflow-related content requests and (ii) the potential drawback of DSF4SP_8 hoarding (transient) replicas around nodes that are capable of executing workflows combined with (iii) the applied replacement strategies not taking into account similarity of content and node identifiers used by the routing concept of the overlay network (see Section 5.4.2.3). This effect is further increased by DSF4SP_6 by another 11.4%, due to the tremendous growth of the average number of transient replicas in the network caused by the latter. DSF4SP_10 again almost achieves the level of content availability observed for DSF4SP_8. Hence, by means of content-class-driven placement of replicas in the content repository set of primary providers, DSF4SP_10 is able to compensate the decrease of content availability caused by the lazy removal properties of transient replicas.

Properties of Replicas and Transient Replicas. The applicability of the strategies in the domain of smart products very much depends on the transient replica and replica ratios. Thus, compared to DSF4SP_8, while DSF4SP_6 almost halves the total number of created transient replicas and reduces the pre-replication failure by approximately 8%, it leads to a significant increase in the average number of transient replicas. The absolute difference between DSF4SP_6 and DSF4SP_8 approximates a value of 12,230 transient replicas. Furthermore, DSF4SP_10 increases the total number of created transient replicas by 22.7% compared to DSF4SP_8. Concerning the average number of transient replicas it even increases DSF4SP_6 by another 13,140 transient replicas. Yet, it is important to note that transient replicas aim at effectively "filling up" unused storage capacity of smart products and do not imply an increase of the number of content replacements (of course depending on the configuration of the related TTL intervals). Given the defined replacement strategies, only "persistent" replicas lead to a continuous use of storage resources and require content replacement if available storage capacity becomes rare. Concerning the average number of replicas in the network, both DSF4SP_6 and DSF4SP_8 yield very close results that only differ by 3.3%. In contrast, DSF4SP_10 increases the latter by almost 28,000 replicas. Especially in resource-constrained smart product systems, this can represent a considerable burden.

Recommendation. As expected, there is no strategy assembly that fits all potential smart product scenarios and – consequently – there cannot be a single recommendation. Also, the four proposed configurations strongly affect the impact of the content placement strategies. A detailed analysis of the impact of the configurations on (i) the average number of (transient) replicas in the network as well as (ii) the improvement of query efficiency both overall and workflow-related is presented in Section 5.4.2.2.

If resources are limited and query efficiency as well as content availability are to be addressed, than DSF4SP_8 represents the strategy of choice. One could even think

of extending the strategy with a replication concept that maintains a minimal number of replicas on-board of content repositories of primary providers in order to enhance content availability given node churn. Yet, if the communication overhead of CPA represented a limiting factor, then PA or MPP should be used. In particular, the risk-averse pre-replication concept of MPP nicely fits resource-limited nodes targeting workflow execution.

If query efficiency is more important than content availability, then one should opt for DSF4SP_6. DSF4SP_6 enhances query efficiency considerable over DSF4SP_8 by maintaining higher transient replica ratios, only. This might in particular be relevant for scenarios in which content availability can be guaranteed via cloud-based providers and it is primarily workflow execution performance that is to be optimised.

Finally, for smart product systems for which resource limitation is not so much of an issue, it is clearly the strategy DSF4SP_10 that should be used. DSF4SP_10 is able to compensate the impact of the high amount of transient replicas on overall content availability. Even further, it improves workflow-related query efficiency compared to DSF4SP_6 and yields the by far best results in terms of overall query efficiency. This effect is even expected to augment in scenarios and applications that foster the use of content classes (e.g., applications assigning content class *CC_QUERY_EFFICIENCY* also requesting the content repeatedly in a short time). Also, in case content availability becomes more critical, one could think of an adaptation of the replacement strategy ECR that excludes the use of lazy removal properties of transient replicas and focusses on content classes, only.

5.5 Chapter Summary

This chapter described the setup and the results of the simulation-based evaluation study used for assessing the suitability of the proposed content placement strategies in the domain of smart products as well as the actual impact of the strategies on query efficiency and content availability. Based on the information presented in Chapters 2 and 4, this chapter includes a description of the conceptual simulation model (see Section 5.2), a description of the implemented simulation model (see Section 5.3), as well as an analysis of the expected results combined with an interpretation of the simulation data (see Section 5.4).

The evaluation results revealed the anticipated added value of the proposed content placement strategies. Thus, the workflow-based replication strategies combined with the concept of transient replicas improve query efficiency of workflow-related content requests considerably without generating high replica volumes. According to the simulation data, the proposed workflow-based replication strategies could enhance workflow-

related and overall query efficiency by up to 40% and 12% (averaged configurations without taking into account content-class-driven strategies) compared to the baseline given by the assembly NO REPLICATION.

Also, the benefit of more optimistically trading transient replica ratios and pre-replication failure for enhancement of query efficiency (PA and CPA compared to MPP), as well as the value of cooperative knowledge incorporated by CPA was confirmed by the evaluation results. However, the simulation results revealed that (i) the benefit of the cooperative concept of CPA decreases with increasing transient replica ratios (PA slightly outperforms CPA with an incorporation of MFRTR) and (ii) the difference between the proposed strategies gets reduced with an increasing number of (transient) replicas.

The positive effect of the three workflow-based replication strategies is further augmented by the replacement strategy MFRTR, which leverages lazy removal properties of transient replicas. Thus, compared to the baseline given by NO REPLICATION, improvements of up to 59% and 21% were measured for workflow-related and overall query efficiency, respectively (averaged configurations without taking into account content-class-driven strategies). Compared to the best related work (CACHING_3), an improvement of approximately 9% was measured for workflow-related query efficiency. Yet, MFRTR leads to an increased number of transient replicas placed on the requester side, which was observed to negatively affect content availability.

This issue of MFRTR is addressed by the content-class-driven strategies CC and ECR, which achieve the level of content availability measured for workflow-based replication strategies and MFR. Combined with the workflow-based replication strategies, CC and ECR yield the best results in terms of workflow-related and overall query efficiency with an improvement of up to 63% and 28%, respectively, compared to NO REPLICATION.

Overall, given the resource limitation of smart products, the following recommendation on when to apply which of the proposed replication strategies was derived.

Recommendation

- In scenarios with resource-constrained smart products, the strategy assembly [CPA, MFR] is recommended.
- If the complexity and the communication overhead of CPA represents an issue, then the strategy assembly [MPP, MFR] is recommended.
- If resources of smart products facilitate storage and maintenance of transient replicas, then the strategy assembly [PA, MFRTR] is recommended. Since this assembly has a negative impact on content availability, one should think of incorporating a complementary strategy that focusses on content availability.

- If resources of smart products are not an issue, then [CC, CPA, ECR] is the strategy assembly of choice.
- In scenarios with heterogeneous smart products, a resource-dependent combination of the proposed content placement strategies is recommended.

Chapter 6
Conclusion

This chapter concludes the thesis by summarising its contributions and pointing out limitations to be addressed by future work. It is structured as follows. First, Section 6.1 recaps the research question and summarises the proposed content placement strategies. Moreover, the main results of the simulation-based evaluation study as well as shortcomings of the closest related work are presented. Thereafter, Section 6.2 discusses the limitations of the proposed strategies and presents an outlook on future work.

6.1 Contribution

Smart products target simplicity of product use by means of multi-modal and personalised product-to-user interaction as well as active user guidance. For this purpose, smart products are equipped with knowledge and knowledge-related functionality. Amongst others, this includes domain and problem-solving knowledge used by smart products to actively approach and assist their users.

This thesis studies how procedural problem-solving knowledge associated with smart products can be utilised by content placement strategies, i.e., content replication and replacement strategies, to enhance content access. According to the results of the EU-funded research project SmartProducts, the thesis assumes procedural problem-solving knowledge to be modelled in the form of workflows that reflect sequences of activities and feature annotation of activity-related content needs. Also, it assumes complex-structured workflows that consist of multiple alternative branches with the actual branch(es) to be followed being determined dynamically. On this basis, three workflow-based replication strategies as well as a content-class-driven replication strategy are proposed. Also, for each of these approaches, an integrated replacement strategy is defined to capture the complete replication life cycle.

Workflow-based Content Placement. The three workflow-based replication strategies Most Probable Path (MPP), Path Assessment (PA), and Cooperative Path Assessment (CPA) make use of workflow structures to predict upcoming workflow-related content requests. They decouple the control flow and the data flow of workflows, and pre-replicate content according to expected demand in order to enhance query efficiency during workflow processing. MPP relies on past workflow executions in order to determine the workflow branch with the highest probability of being processed. In contrast, PA and CPA more optimistically balance enhancement of query efficiency with pre-replication wastage. They calculate pre-replication benefit and cost for all activity-related content needs and assess workflow branches with a dedicated metric. This way, PA and CPA may pre-replicate content needs for multiple paths. Even further, while MPP and PA make replication decisions based on local knowledge, CPA is a cooperative strategy that takes into account content interest of surrounding smart products.

All three strategies make use of a configurable pre-replication reach, which is adjusted to control flow progress in order to limit pre-replication wastage. Moreover, they apply the proposed concept of transient replicas. Thus, pre-replicated content results in transient replicas with pre-defined gradually increasing TTL intervals. While transient replicas may be persisted in case the number of requests exceeds a pre-defined threshold, they are removed in case they are not accessed within their active TTL. This avoids pre-replication wastage in the long term and limits the number of replicas maintained in the distributed storage system.

The three workflow-based replication strategies are complemented by the replacement policy MFR for Transient Replicas (MFRTR), which extends the replacement policy MFR [KRT06, KRT07] by taking into consideration transient replicas with lazy removal properties. Instead of removing transient replicas that were not accessed during their active TTL, these replicas are flagged as candidates for being removed. While such candidates can be re-initiated in case they are accessed (i.e., the removal flag is revoked), they are removed eventually by MFRTR if storage capacity needs to be made available for serving pending storage requests. If this does not free the required capacity, then MFRTR adopts MFR for identifying content to be replaced given the remaining storage capacity that needs to be made available. Hence, MFRTR enables filling up of unused storage capacity of smart products without increasing replacement overhead.

Content-Class-based Content Placement. The content-class-based replication strategy Content Class (CC) enables smart products to specify the primary property of content they are operating with. For this purpose, the four distinct content classes *CC_PERSISTENCE*, *CC_AVAILABILITY*, *CC_QUERY_EFFICIENCY*, and *CC_DEFAULT* are proposed. CC analyses content class assignments and provides class-specific policies to adapt replica organisation. For example, while content class *CC_PERSISTENCE* leads to erasure-code replication, the class *CC_QUERY_EFFICIENCY* results in replica placement on the requester side to improve potentially upcoming requests. Hence, instead of relying on observed access patterns or optimising replica organisation with regard to workflow operations, this approach provides smart products with an explicit means for specifying their intended content use and enables adaptation of number and placement of replicas, accordingly.

The strategy CC is complemented by the replacement policy Enhanced Content Replacement (ECR), which extends MFRTR by means of a content-class-driven approach for determining replacement candidates. For this purpose, ECR defines a location quality ordering of content classes, namely *CC_DEFAULT* $<$ *CC_PERSISTENCE* $<$ *CC_AVAILABILITY* $<$ *CC_QUERY_EFFICIENCY*. On this basis, ECR tries to replace content with lower location dependency than the content to be stored instead of purely relying on access recency and frequency. If this does not free the required storage capacity, then ECR determines content of the same class as the new content to be stored and applies a cost / benefit metric to determine the added value of storing the new content on-board. Content of higher location quality than the new content is not taken into account. This way, ECR aims at preserving the location properties according to the content-assigned classes given the limited storage capacity of smart products.

Simulation-based Evaluation. The results of the simulation-based evaluation study confirm the anticipated added value of the proposed content placement strategies. MPP, PA, and CPA combined with the concept of transient replicas improve query efficiency of workflow-related content requests considerably without generating high replica volumes. Also, the benefit of more optimistically trading pre-replication wastage for en-

hancement of query efficiency (PA and CPA compared to MPP), as well as the value of the cooperative approach applied by CPA is confirmed by the evaluation results. This effect is further augmented by the replacement strategy MFRTR. Yet, the increased number of transient replicas maintained by MFRTR negatively affects content availability. The content-class-driven strategies CC and ECR address this issue and yield the best results in terms of query efficiency and content availability. However, as opposed to the workflow-based replication strategies, CC and ECR cause a considerable increase in the number of replicas. Also, the results of the active replication strategy MDCDN, which is used as representative for active replication strategies that apply access pattern recognition for estimating upcoming demand, as well as of a caching-like approach could be improved by the proposed content placement strategies for almost all quality attributes considered in the evaluation study.

Conclusion. An analysis of state-of-the-art replication strategies revealed that most replication strategies do not utilise workflow models for pre-replicating content according to predicted upcoming demand. The few existing strategies that follow this approach assume purely sequential workflow structures. Hence, they do not cope with control flow uncertainty and are not applicable in the domain of smart products. The approaches related to the proposed replication strategy CC support a single objective, only. While there are concepts that enable applications to specify query efficiency or content availability targets, there is no generic strategy that supports differentiated content classification and content-class-driven adjustment of replica organisation. In conclusion, as to the knowledge of the author, the proposed content placement strategies are the first that address the investigated requirements while taking into account the challenges of smart products.

6.2 Future Work

Workflow-based Content Placement. The proposed strategies apply a static configuration of the parameters used for controlling pre-replication reach as well as the number of workflow branches for which content is pre-replicated. Given the inherent heterogeneity of smart products, one could think of a dynamic adaptation of these parameters subject to the relation between upcoming content needs and available storage capacity. Also, energy limitations could be taken into account. While resource-constrained smart products could try to limit pre-replication wastage by low values assigned to the parameters, powerful products could assign higher values in order to more optimistically trade pre-replication wastage for improvement of query efficiency. This would also enable PA and CPA to fall back to MPP (by setting the path assessment threshold to a sufficiently high value) as well as all strategies to fall back to traditional on-demand retrieval of activity-related content needs to avoid pre-replication wastage (by setting the schedule look ahead parameter to a value of 0), if needed.

Content-Class-based Content Placement. The strategy CC supports qualitative content categorisation and best effort policies, only. In future work, this could be extended to allow for the definition of quantitative properties such as maximum access latency or specific availability levels known from SLAs. This requires an adaptation of the class-specific policies similar to the concepts proposed by [On04] and [HA04] that target guaranteeing application-assigned levels of content availability and query efficiency, respectively. Yet, it has to be studied whether strategies with quality guarantees fit the limited resources of smart products.

Evaluation Study. Indeed, simulation-based evaluation studies present an initial idea of the expected behaviour of the proposed strategies. Yet, the degree of abstraction of simulation models limits the generic validity of simulation results. This thesis tries to approach this issue by complementing the simulation results with analyses of the expected outcome. Still, future work should extend the evaluation study. First, the simulation model should be enriched by additional (i) underlay network topologies, (ii) overlay network implementations, (iii) implementations of related work, as well as (iv) evaluation scenarios. The growth of the amount of simulation data would provide additional insights by enabling the application of statistical hypotheses tests and would facilitate the formulation of well-founded recommendations on when to apply which strategy. In a second step, the simulation-based evaluation should be complemented by experiments. The results gained from test-bed or pilot experiments provide important insights that could moreover be used to verify the simulation model and enable enhanced simulation of future research.

Consistency Maintenance. This thesis focuses on content placement strategies but abstracts from update operations. In fact, consistency maintenance is a research area on its own. However, in order to define a complete distributed storage system for smart products, the proposed strategies are to be complemented with consistency maintenance concepts. Given the concept of transient replicas, one could think of consistency maintenance mechanisms covering (persistent) replicas, only. This concept is applied by the distributed storage system OceanStore (see Section 3.1.3.1), which distinguishes between primary and secondary (soft state) replicas. Also, one could think of adapting the proposed replication strategies to (i) reduce replication degrees for content with high observed update rates and (ii) limiting pre-replication of workflow branches other than the one with the highest execution probability to content with low updates rates or read-only properties. This could reduce complexity and cost of maintaining content consistency.

Architecture. Finally, while the proposed strategies are not tailored to any specific architecture, one could think of optimising the latter towards hierarchical network structures such as hybrid P2P overlay networks. For example, the proposed workflow-based strategies could be operated by cluster heads, only, in order to enable aggregated assessment of workflow branches. This could foster utilisation of temporal and geographic locality properties and could lead to further optimisation of content access.

Annex

Chapter A
Implementation and Evaluation

A.1 Process Flow of Simulation Events

This section presents an overview of the process flow of the overlay network events *LEAVE* (Figure A.42) and *FAIL* (Figure A.43) as well as the overlay service events *PUT* (Figure A.44), *GET* (Figure A.45), and *WORKFLOW* (Figure A.46). A description of the *JOIN* event is presented in Section 5.3.1.2. While most events are self-explanatory, the assignment of content class made by *PUT* and *GET* events deserve an additional note.

Thus, the process phase of the *GET* event randomly selects a content object of all objects stored in the network following a Zipf-like distribution. The actual assignment of a content class to the *GET* request, i.e., which properties has the content for the requester, is made based on the content class distribution defined for the simulation scenario. This approach is also applied to *PUT* events. Moreover, the latter create new content objects of dynamic size according to the content type linked with the node class of the node for which the *PUT* event has been scheduled.

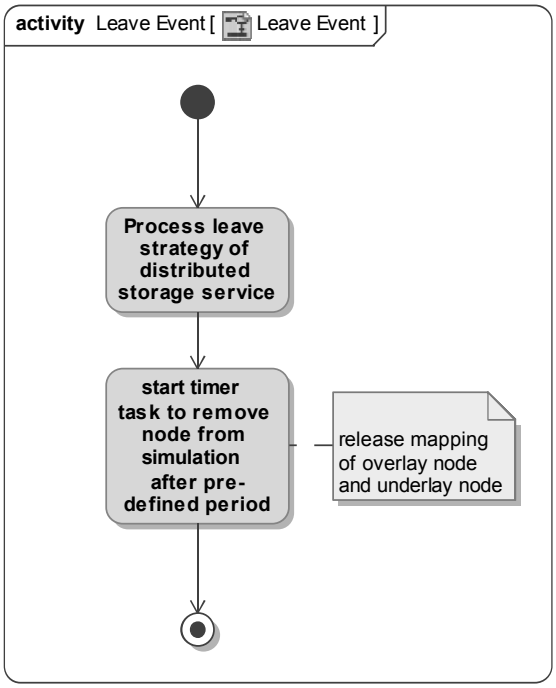


Figure A.42: Process Flow of *LEAVE* Event

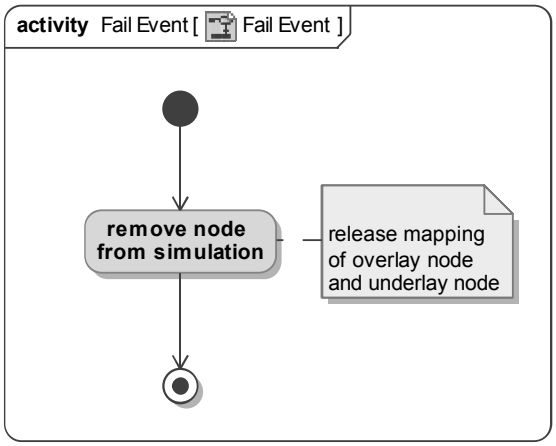


Figure A.43: Process Flow of *FAIL* Event

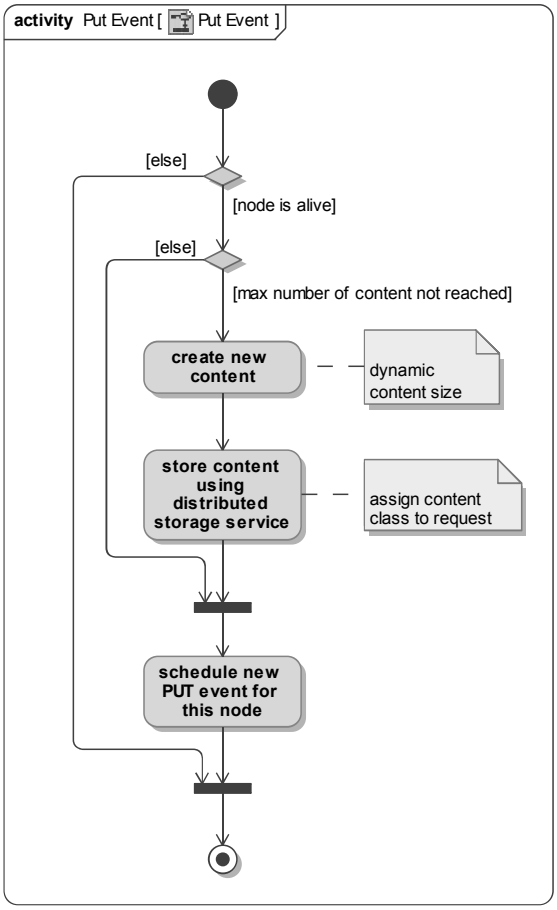


Figure A.44: Process Flow of *PUT* Event

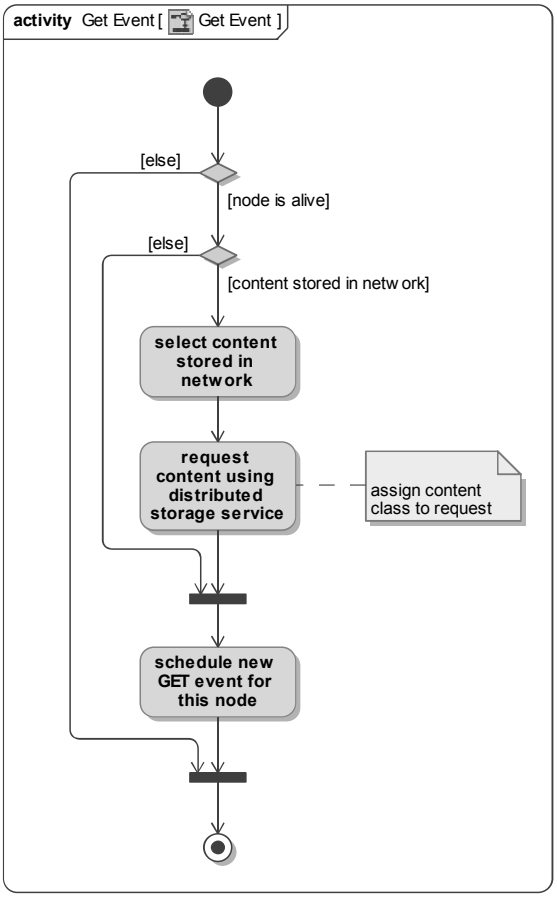


Figure A.45: Process Flow of *GET* Event

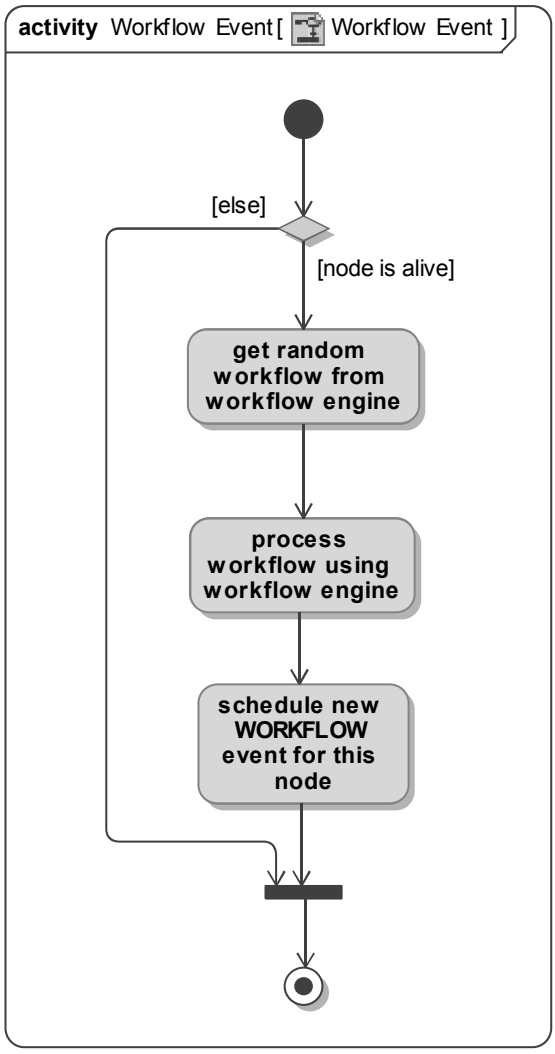


Figure A.46: Process Flow of *WORKFLOW* Event

A.2 Underlay Network

A.2.1 Configuration of BRITE Topology Generator

The following Figures A.47, A.48, and A.49 present the configuration made in the BRITE topology generator to meet the network topology properties described in Section 5.2.3.

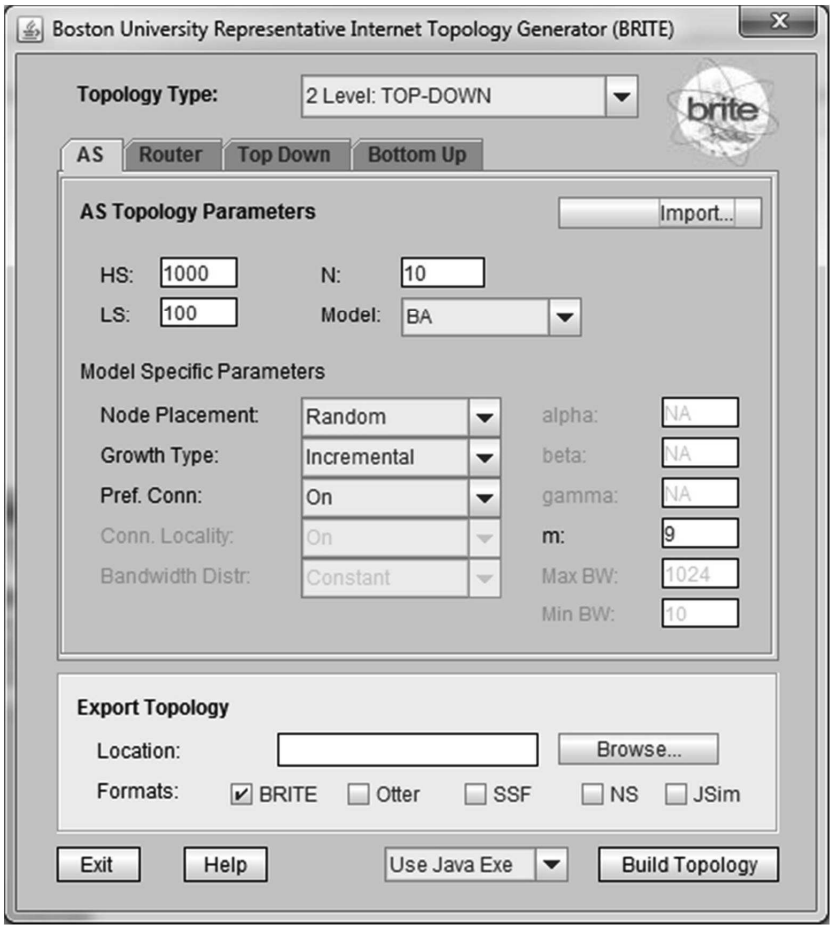


Figure A.47: BRITE Topology Generator Configuration (1/3)

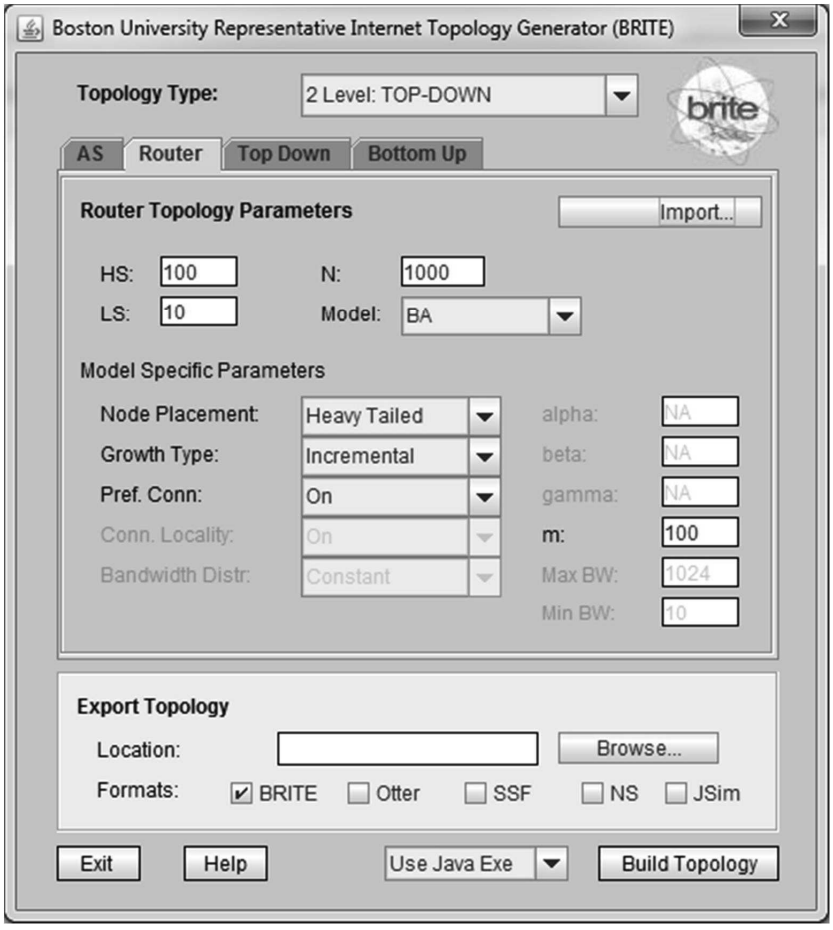


Figure A.48: BRITE Topology Generator Configuration (2/3)

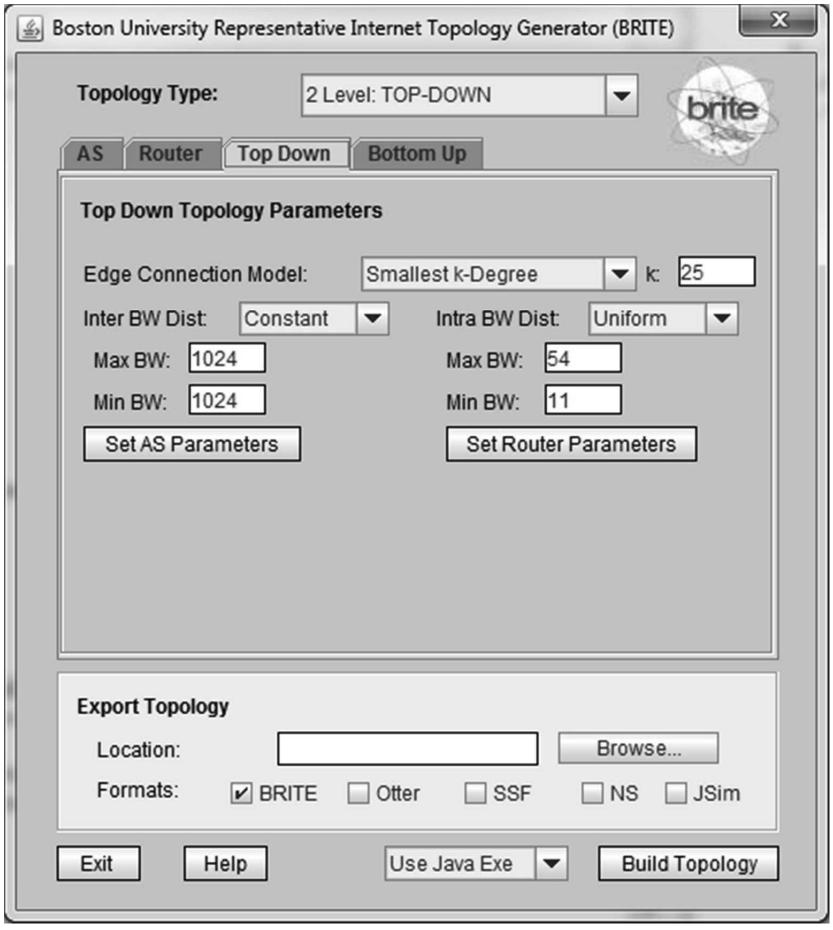


Figure A.49: BRITE Topology Generator Configuration (3/3)

A.2.2 Generated BRITE Topology

Figure A.50 presents a snapshot of the network topology generated according to the configuration presented in Section A.2.1.

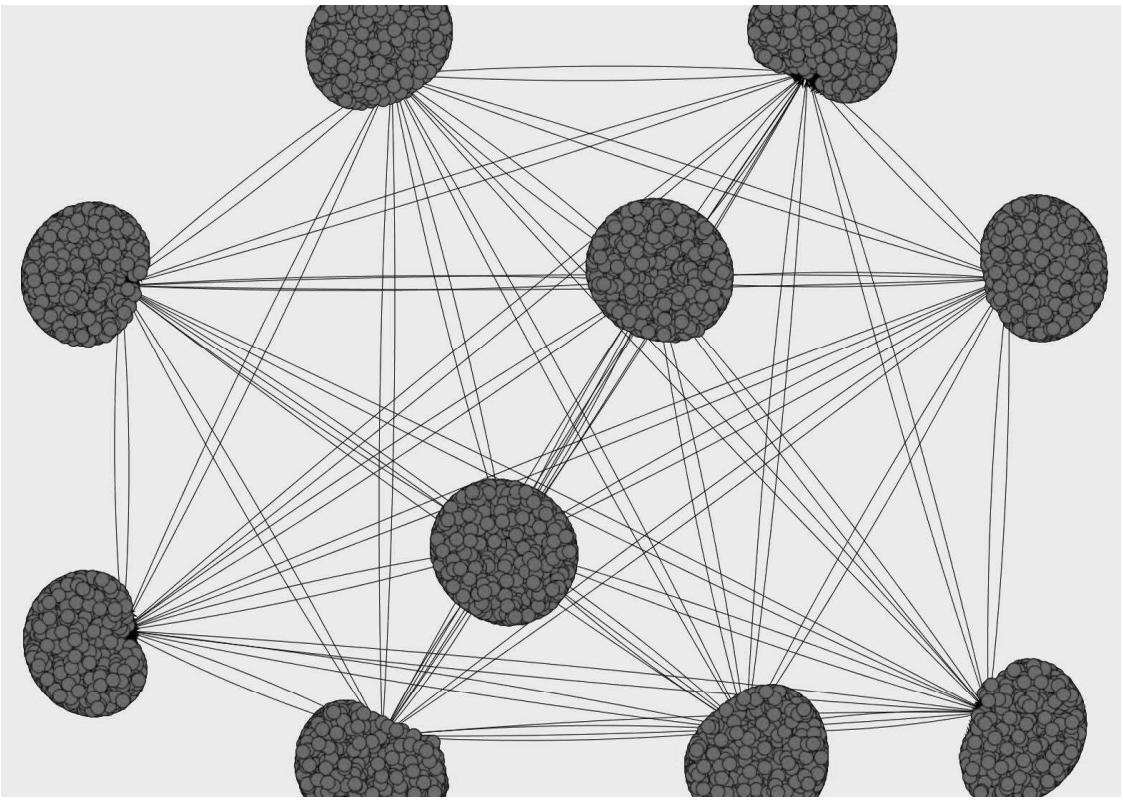


Figure A.50: Implemented Simulation Model: Underlay Network Topology

A.3 Distributed Storage Service

This section presents the software structure of the main components of the distributed storage service by means of class diagrams.

A.3.1 Storage Strategy

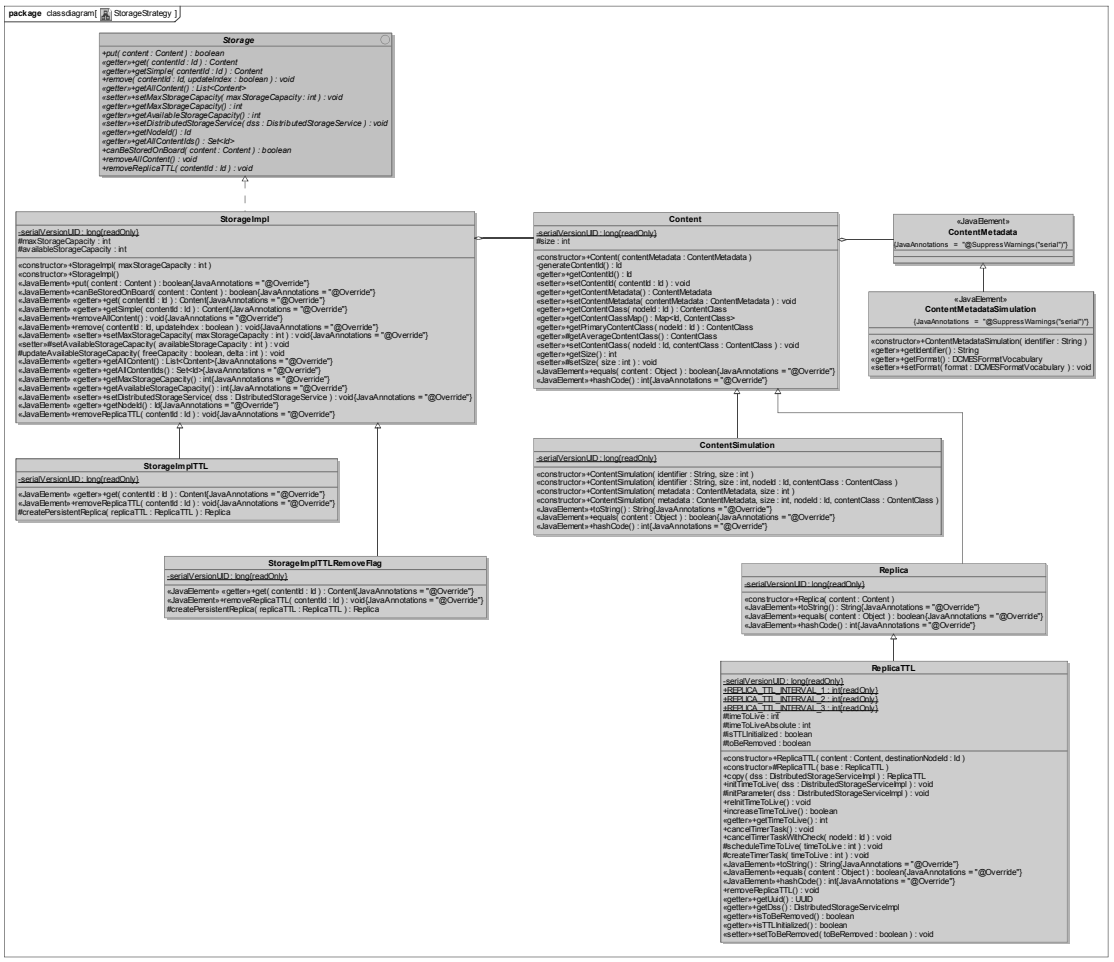


Figure A.51: Distributed Storage Service: Class Diagram of Storage Strategy

A.3.2 Index Strategy

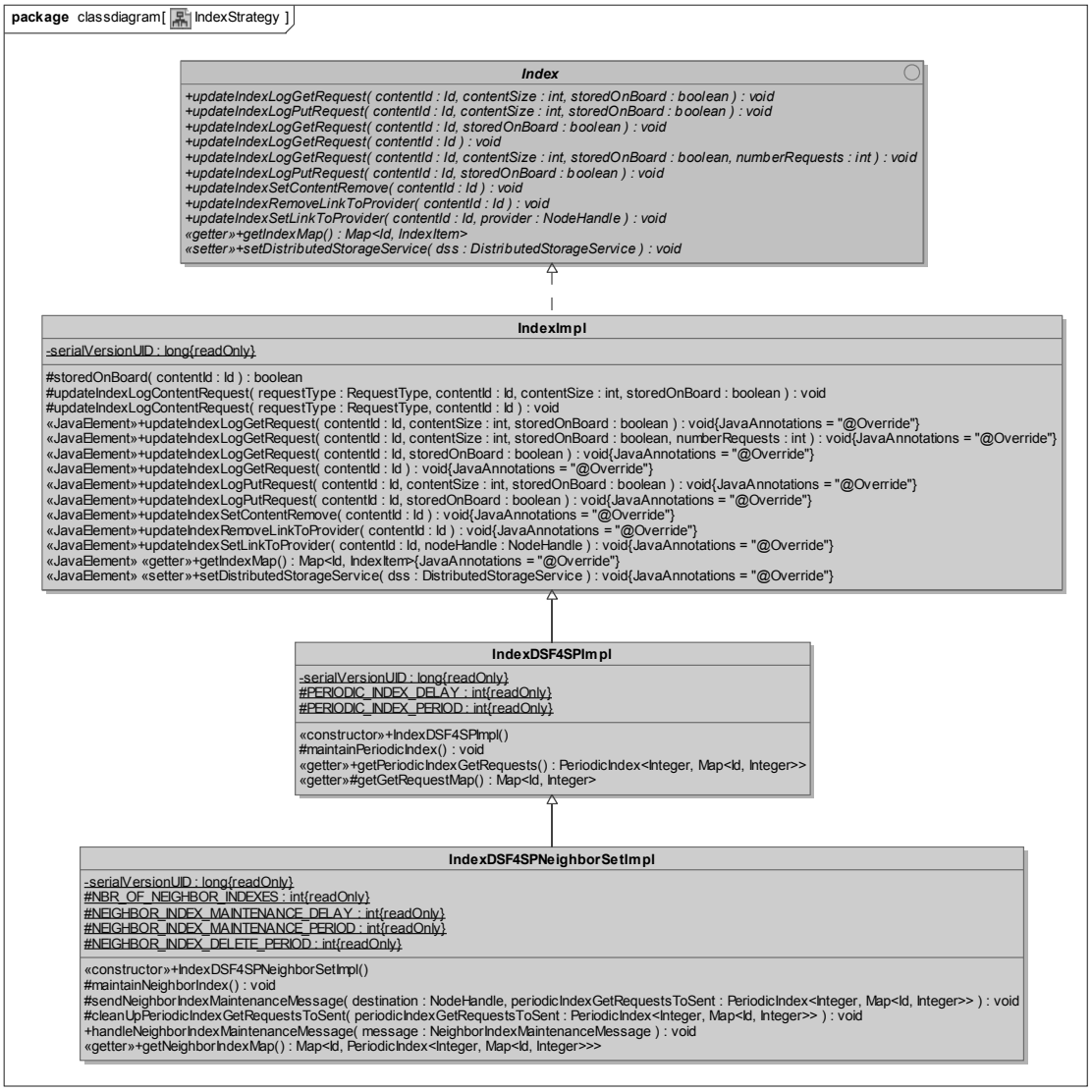


Figure A.52: Distributed Storage Service: Class Diagram of Index Strategy

A.3.3 Get Strategy

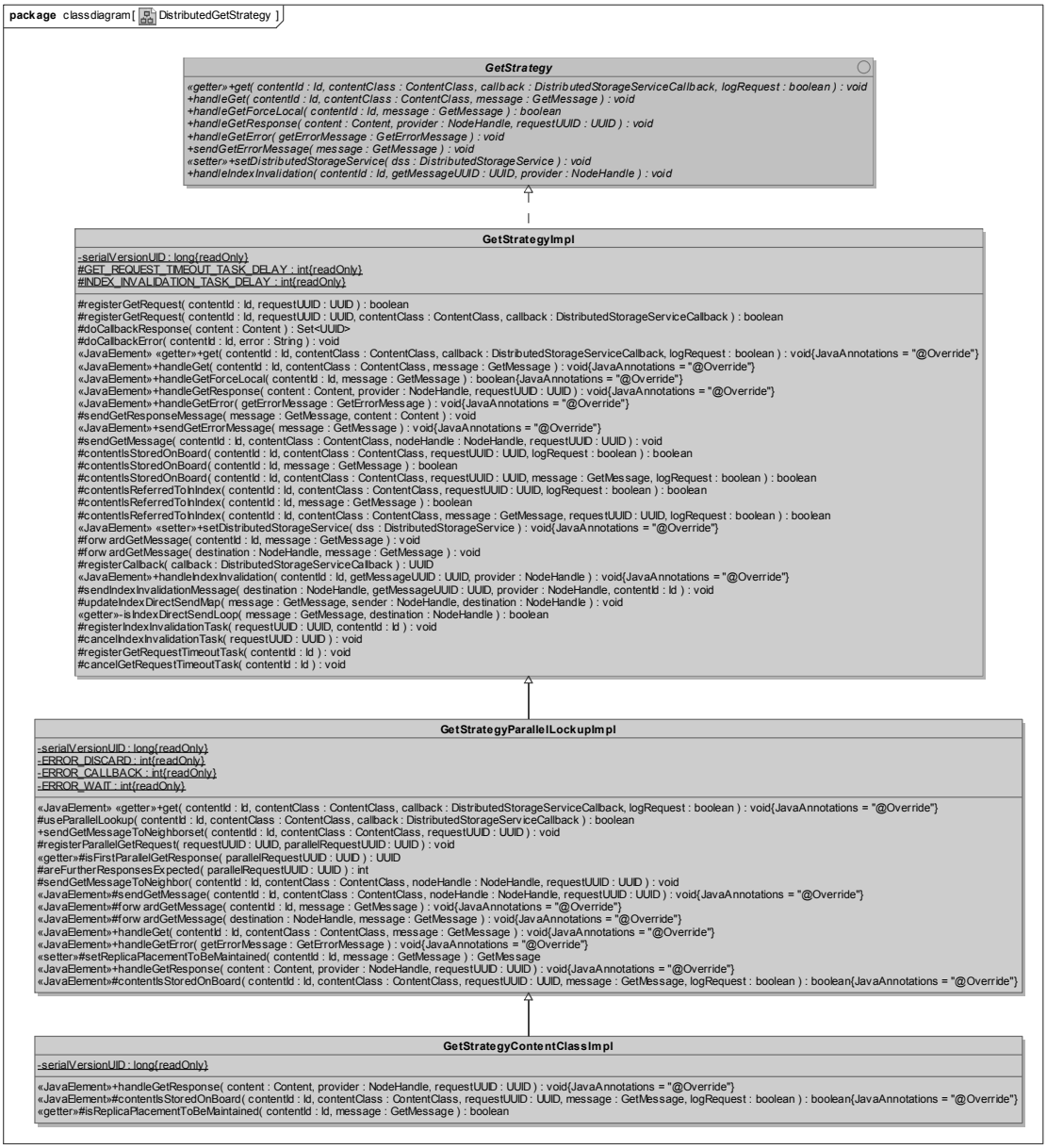


Figure A.53: Distributed Storage Service: Class Diagram of Get Strategy

A.3.4 Put Strategy

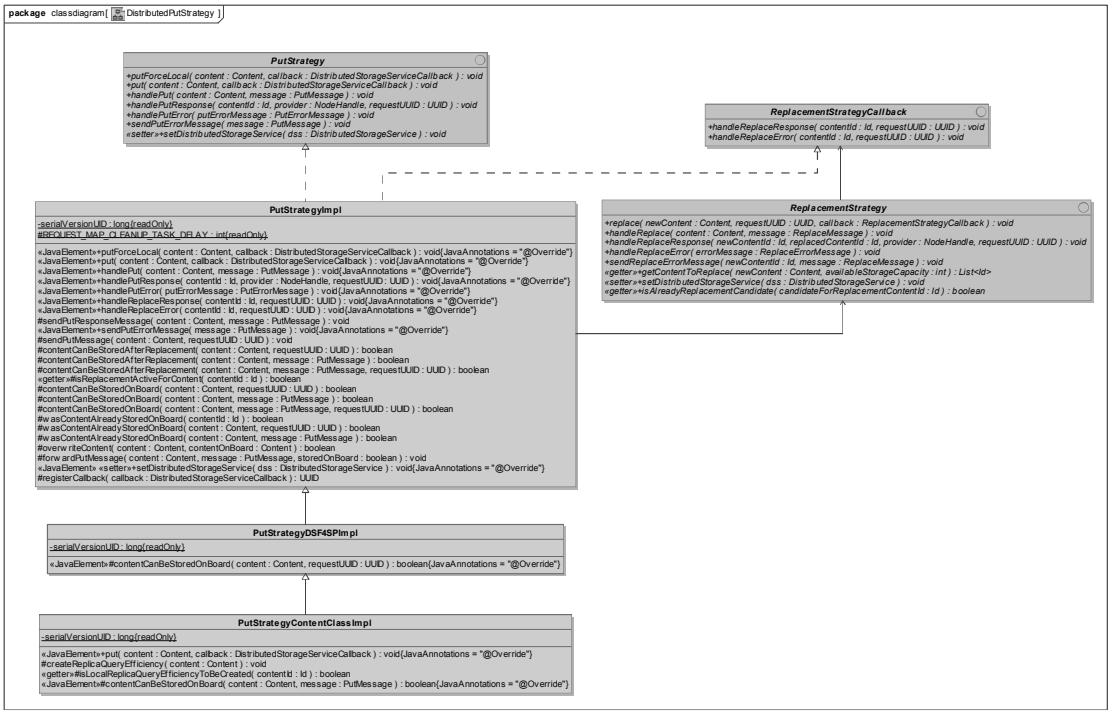


Figure A.54: Distributed Storage Service: Class Diagram of Put Strategy

A.3.5 Replacement Strategy

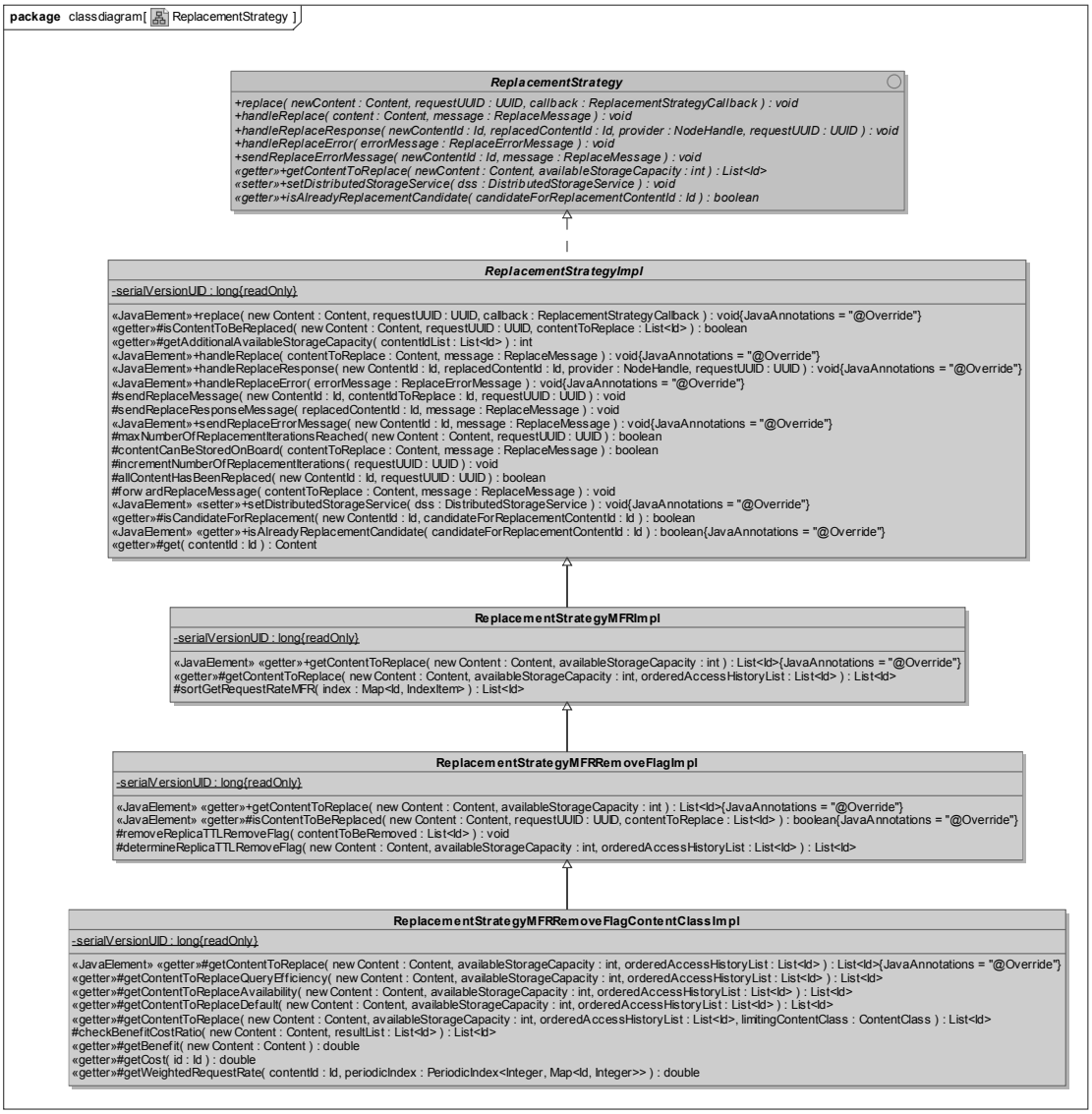


Figure A.55: Distributed Storage Service: Class Diagram of Replacement Strategy

A.3.6 Replication Strategy

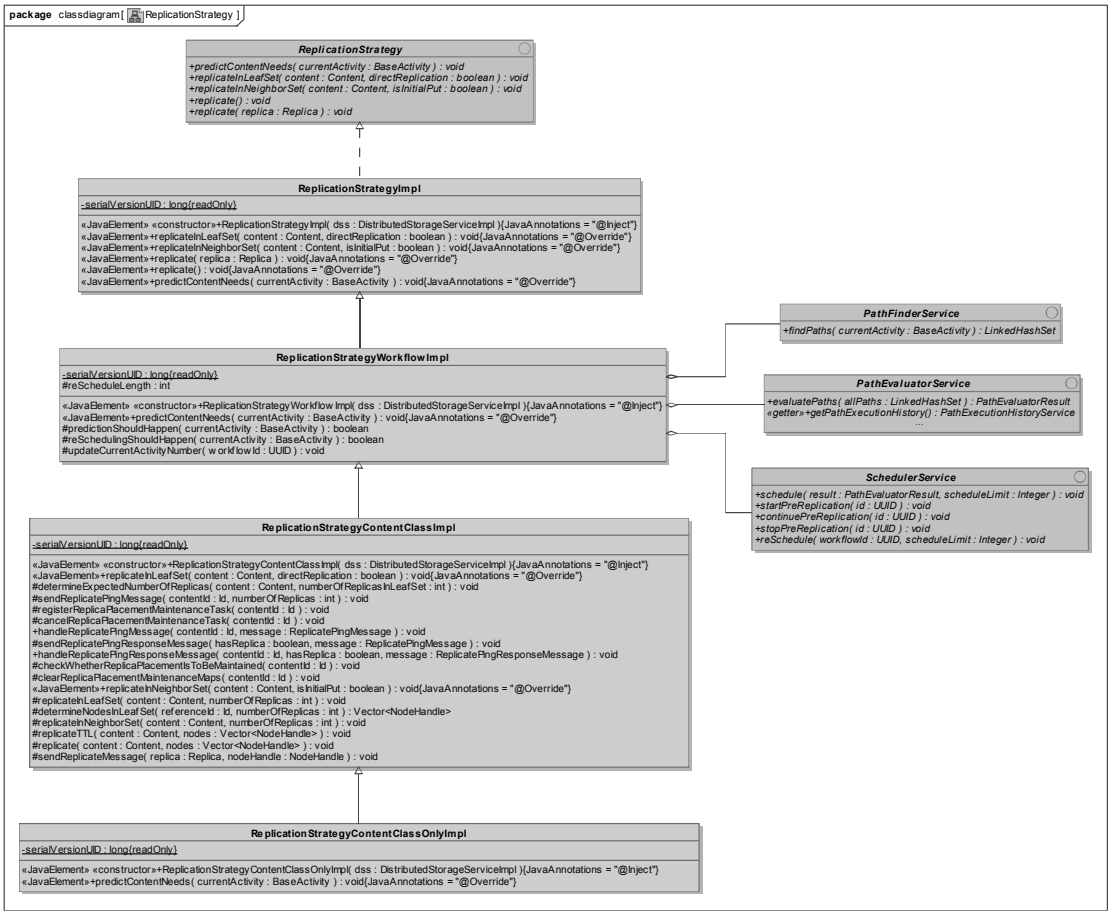


Figure A.56: Distributed Storage Service: Class Diagram of Replication Strategy (1/2)

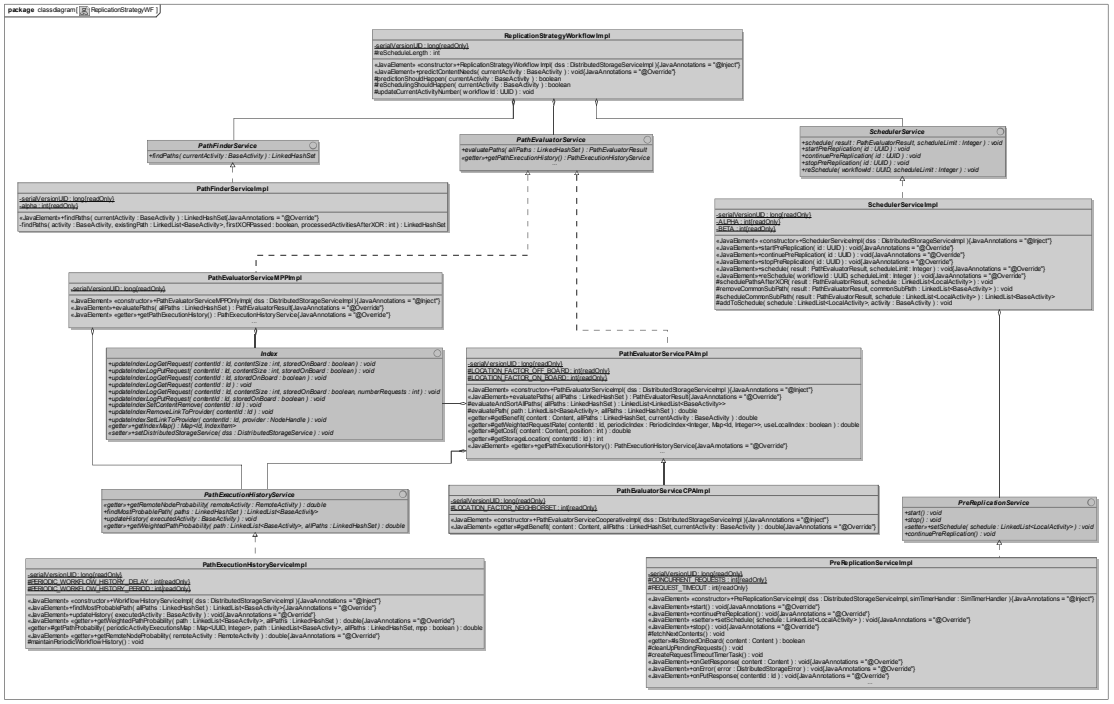


Figure A.57: Distributed Storage Service: Class Diagram of Replication Strategy (2/2)

A.4 Analysis of Simulation Data

A.4.1 Simulation Data

Tables A.32 and A.33 present the results of the evaluation study. The column headers refer to quality attributes summarised in Table 5.14. Note that all quality attributes but $\{QA4, QA5, QA6\}$ depict average values calculated as arithmetic mean values. $QA2.2$ and $QA2.3$ represent the average number of replicas and transient replicas in the network, respectively. $QA6.1$ and $QA6.2$ show the transient replica utilisation in absolute and relative form. The two attributes $QA1$ and $QA3$ are not covered by the table, because they are configuration-independent. $QA1$ is shown in the main evaluation study (see Section 5.4.2). Also, the average number of content in the network is not represented, because is can be derived from $QA2.2, QA2.3$ (remember that the strategies do not cover any means for content removal).

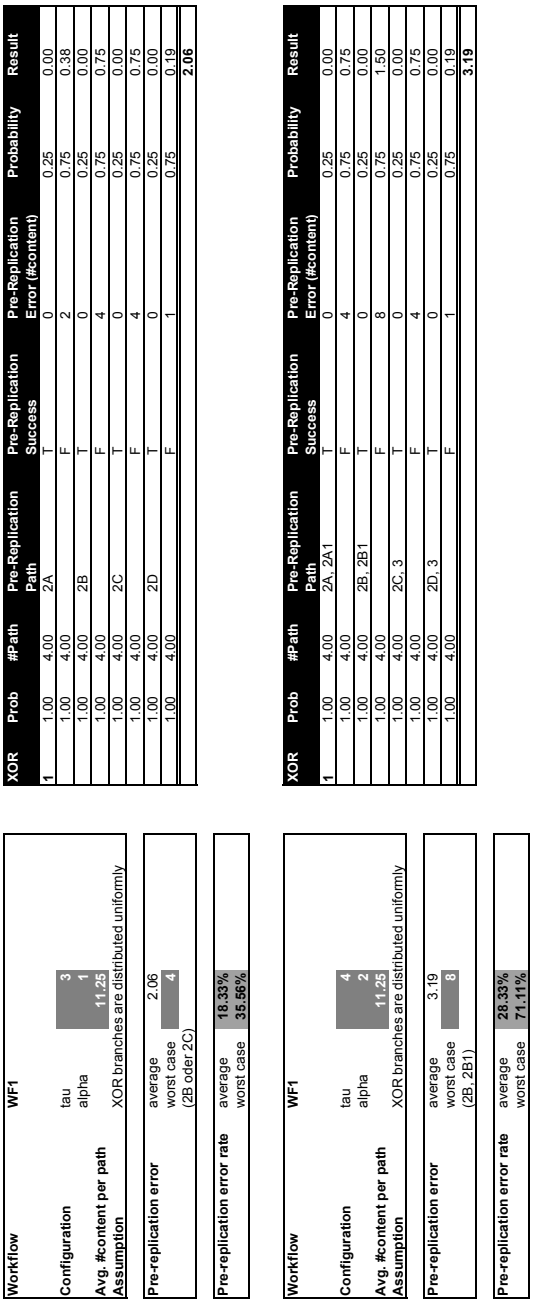
	QA2.2	QA2.3	QA4	QA5	QA6.1 (0)	QA6.1 (1)	QA6.1 (2)	QA6.1 (3)	QA6.2 (0)	QA6.2 (1)	QA6.2 (2)
DSF4SP_1_3	20988.21	8813.18	66.23%	27.81%	30693	414	87	143	97.94%	1.32%	0.28%
DSF4SP_1_4	34771.93	17793.51	105.56%	54.02%	51443	1813	354	519	95.04%	3.35%	0.65%
DSF4SP_2_1	84.98	297.07	0.27%	0.96%	6278	38102	661	172	13.89%	84.27%	1.46%
DSF4SP_2_2	128.91	387.52	0.41%	1.25%	5477	37988	740	242	12.32%	85.47%	1.66%
DSF4SP_2_3	146.15	561.89	0.47%	1.80%	11249	45312	776	276	19.53%	78.65%	1.35%
DSF4SP_2_4	177.74	732.73	0.57%	2.37%	9984	45547	951	334	17.57%	80.17%	1.67%
DSF4SP_3_1	93.48	10842.94	0.30%	34.83%	2457	17279	304	178	12.15%	85.46%	1.50%
DSF4SP_3_2	118.37	10791.71	0.38%	34.82%	2001	17301	370	213	10.06%	87.01%	1.86%
DSF4SP_3_3	129.66	14092.09	0.42%	45.31%	4624	20576	321	246	17.95%	79.85%	1.25%
DSF4SP_3_4	175.84	14367.19	0.56%	46.15%	4367	21490	396	327	16.43%	80.85%	1.49%
DSF4SP_4_1	21158.12	19752.85	66.84%	62.40%	32485	19008	297	194	62.49%	36.57%	0.57%
DSF4SP_4_2	35013.66	27013.02	105.78%	81.61%	47410	20055	457	327	69.47%	29.39%	0.67%
DSF4SP_4_3	21322.40	24684.16	67.43%	78.06%	36720	23707	381	273	60.12%	38.81%	0.62%
DSF4SP_4_4	34788.33	31592.41	105.14%	95.48%	48231	24930	505	396	65.12%	33.66%	0.68%
DSF4SP_5_1	83.45	298.11	0.27%	0.96%	7134	37789	660	161	15.60%	82.61%	1.44%
DSF4SP_5_2	132.46	393.27	0.43%	1.26%	6190	38068	779	248	13.67%	84.06%	1.72%
DSF4SP_5_3	133.79	582.72	0.43%	1.88%	12943	46148	831	269	21.50%	76.67%	1.38%
DSF4SP_5_4	185.59	755.92	0.60%	2.44%	12185	46089	916	358	20.46%	77.40%	1.54%
DSF4SP_6_1	92.63	10861.09	0.30%	35.07%	2765	17158	285	182	13.56%	84.15%	1.40%
DSF4SP_6_2	127.61	10908.22	0.41%	35.16%	2316	17264	347	227	11.49%	85.66%	1.72%
DSF4SP_6_3	141.75	14724.21	0.45%	47.02%	5566	20949	324	251	20.55%	77.33%	1.20%
DSF4SP_6_4	184.56	14450.42	0.60%	46.59%	5014	20773	416	332	18.90%	78.29%	1.57%
DSF4SP_7_1	21210.25	20244.75	66.87%	63.83%	34548	19062	302	186	63.86%	35.24%	0.56%
DSF4SP_7_2	34908.53	28001.61	106.03%	85.05%	48296	20201	413	364	69.72%	29.16%	0.60%
DSF4SP_7_3	21210.26	24460.24	66.57%	76.77%	36986	23345	385	266	60.65%	38.28%	0.63%
DSF4SP_7_4	34889.15	31498.92	104.72%	94.54%	49122	24484	513	426	65.90%	32.84%	0.69%
DSF4SP_8_1	108.01	291.63	0.35%	0.94%	6922	36970	610	195	15.49%	82.71%	1.36%
DSF4SP_8_2	122.36	392.56	0.39%	1.26%	5960	38163	841	242	13.18%	84.42%	1.86%
DSF4SP_8_3	153.68	579.75	0.49%	1.85%	12903	45817	852	284	21.56%	76.55%	1.42%
DSF4SP_8_4	181.13	752.33	0.58%	2.42%	11998	45858	929	339	20.29%	77.56%	1.57%
DSF4SP_9_1	90.06	11045.75	0.29%	35.23%	2863	17039	299	168	14.06%	83.65%	1.47%
DSF4SP_9_2	119.08	10869.61	0.38%	34.87%	2304	17492	343	231	11.31%	85.87%	1.68%
DSF4SP_9_3	136.52	14293.61	0.44%	45.78%	5475	20304	361	245	20.75%	76.95%	1.37%
DSF4SP_9_4	187.57	14660.83	0.60%	47.00%	5016	21208	422	336	18.59%	78.60%	1.56%
DSF4SP_10_1	21401.03	20327.28	67.27%	63.90%	34042	19386	347	178	63.10%	35.93%	0.64%
DSF4SP_10_2	34842.94	27080.87	105.13%	81.71%	46855	20045	473	334	69.20%	29.61%	0.70%
DSF4SP_10_3	21277.54	24434.51	67.13%	77.09%	36477	23351	407	267	60.29%	38.60%	0.67%
DSF4SP_10_4	35026.21	31670.02	105.91%	95.77%	48866	24317	487	405	65.97%	32.83%	0.66%
MDCDN_1	1773.47	0.00	5.68%	0.00%	0	0	0	0	0.00%	0.00%	0.00%
MDCDN_2	888.46	0.00	2.86%	0.00%	0	0	0	0	0.00%	0.00%	0.00%
CACHING_1	6282.76	0.00	20.11%	0.00%	0	0	0	0	0.00%	0.00%	0.00%
CACHING_2	1087.33	0.00	3.49%	0.00%	0	0	0	0	0.00%	0.00%	0.00%
CACHING_3	27281.61	0.00	83.15%	0.00%	0	0	0	0	0.00%	0.00%	0.00%
NOREP	0.00	0.00	0.00%	0.00%	0	0	0	0	0.00%	0.00%	0.00%

Table A.32: Simulation Data (1/2)

	QA6.2 (3)	QA7	QA9	QA8	QA10	QA12	QA11	QA13	QA14	QA15	QA16
DSF4SP_1_3	0.46%	28.18	27.03	25.23	34.40	32.97	31.26	70.34%	72.09%	59.25%	57.82%
DSF4SP_1_4	0.96%	25.93	23.58	20.49	35.15	31.57	27.97	70.44%	72.38%	52.78%	52.55%
DSF4SP_2_1	0.38%	28.50	25.97	20.95	31.48	28.97	24.35	71.43%	73.65%	72.67%	69.04%
DSF4SP_2_2	0.54%	28.21	25.65	20.28	31.48	28.82	23.70	71.05%	73.26%	70.90%	67.97%
DSF4SP_2_3	0.48%	26.21	22.49	14.60	29.39	25.41	17.41	70.14%	72.67%	68.86%	67.02%
DSF4SP_2_4	0.59%	25.82	22.10	14.02	29.06	25.01	16.71	69.99%	72.76%	67.79%	66.68%
DSF4SP_3_1	0.88%	25.95	22.37	15.36	35.23	30.76	22.04	63.52%	66.72%	55.65%	54.72%
DSF4SP_3_2	1.07%	25.58	21.88	15.02	34.35	29.58	21.28	62.44%	65.50%	54.62%	53.56%
DSF4SP_3_3	0.95%	23.50	18.67	9.09	32.01	25.46	12.99	62.00%	64.87%	52.94%	52.11%
DSF4SP_3_4	1.23%	23.59	18.71	9.25	32.24	25.72	13.31	63.02%	65.44%	53.35%	52.41%
DSF4SP_4_1	0.37%	23.63	20.45	13.94	34.36	30.12	21.50	68.72%	70.68%	50.84%	50.66%
DSF4SP_4_2	0.48%	22.59	19.18	12.78	33.32	28.11	19.48	69.40%	71.02%	48.67%	48.57%
DSF4SP_4_3	0.45%	21.65	17.15	8.26	31.85	25.47	12.87	69.17%	70.70%	49.99%	49.98%
DSF4SP_4_4	0.53%	20.84	16.26	7.59	31.26	24.30	11.76	70.62%	71.76%	48.40%	48.46%
DSF4SP_5_1	0.35%	28.80	26.24	21.05	31.87	29.26	24.46	70.63%	72.98%	72.00%	68.20%
DSF4SP_5_2	0.55%	27.91	25.14	19.69	31.03	28.08	22.79	71.27%	73.79%	71.09%	68.50%
DSF4SP_5_3	0.45%	25.99	22.15	13.89	28.95	24.83	16.40	70.59%	72.76%	69.44%	67.26%
DSF4SP_5_4	0.60%	25.75	21.69	13.42	28.97	24.53	16.05	70.81%	73.45%	68.54%	67.32%
DSF4SP_6_1	0.89%	25.96	22.39	15.48	34.49	30.14	21.92	62.30%	65.37%	54.92%	53.70%
DSF4SP_6_2	1.13%	25.63	21.77	14.82	34.30	29.19	20.65	62.27%	65.20%	54.45%	53.25%
DSF4SP_6_3	0.93%	23.25	18.32	8.64	31.53	24.92	12.34	63.08%	65.73%	53.51%	52.74%
DSF4SP_6_4	1.25%	23.53	18.32	8.74	32.24	25.23	12.54	62.29%	64.97%	52.85%	52.25%
DSF4SP_7_1	0.34%	23.43	20.31	13.90	33.93	29.79	21.29	68.46%	70.16%	50.58%	50.18%
DSF4SP_7_2	0.53%	22.67	19.19	12.81	33.37	28.15	19.45	69.50%	71.18%	48.71%	48.72%
DSF4SP_7_3	0.44%	21.62	16.98	8.01	32.04	25.28	12.47	68.70%	70.16%	49.63%	49.59%
DSF4SP_7_4	0.57%	20.77	16.14	7.35	31.08	24.04	11.43	70.52%	72.22%	48.36%	48.83%
DSF4SP_8_1	0.44%	28.39	25.85	20.83	31.31	28.71	23.96	69.65%	72.15%	70.94%	67.43%
DSF4SP_8_2	0.54%	28.28	25.55	20.22	31.66	28.77	23.78	71.38%	73.69%	71.26%	68.41%
DSF4SP_8_3	0.47%	25.95	22.03	13.85	29.16	25.02	16.57	70.75%	73.31%	69.13%	67.34%
DSF4SP_8_4	0.57%	25.71	21.61	13.47	29.07	24.53	16.09	70.19%	72.76%	67.94%	66.47%
DSF4SP_9_1	0.82%	25.31	21.74	14.75	33.73	29.28	20.80	63.06%	66.71%	55.20%	54.65%
DSF4SP_9_2	1.13%	25.76	22.06	15.10	34.61	29.84	21.50	62.51%	65.69%	54.67%	53.92%
DSF4SP_9_3	0.93%	23.40	18.43	8.89	32.09	25.49	12.87	62.41%	65.20%	53.02%	52.28%
DSF4SP_9_4	1.25%	23.09	18.01	8.47	31.71	24.73	12.18	62.63%	64.66%	52.85%	51.60%
DSF4SP_10_1	0.33%	23.70	20.55	14.05	34.50	30.35	21.67	69.47%	71.26%	51.02%	51.02%
DSF4SP_10_2	0.49%	22.84	19.31	12.79	33.68	28.29	19.58	69.87%	71.39%	49.06%	48.98%
DSF4SP_10_3	0.44%	21.11	16.79	7.77	31.02	24.73	12.00	69.50%	70.58%	49.96%	49.60%
DSF4SP_10_4	0.55%	21.09	16.35	7.50	31.79	24.50	11.69	70.37%	71.68%	48.27%	48.36%
MDCDN_1	0.00%	30.12	28.94	26.76	33.55	32.41	30.64	70.81%	73.73%	69.01%	66.27%
MDCDN_2	0.00%	30.97	29.79	27.39	33.72	32.59	30.34	71.55%	73.97%	72.56%	68.39%
CACHING_1	0.00%	27.35	24.96	20.38	31.56	28.86	24.02	68.99%	71.47%	61.37%	59.83%
CACHING_2	0.00%	30.03	28.44	25.71	32.61	30.81	28.16	70.66%	73.21%	70.36%	66.96%
CACHING_3	0.00%	23.41	19.79	13.01	29.38	24.88	16.85	67.94%	69.73%	51.16%	50.79%
NOREP	0.00%	30.89	30.23	28.75	30.89	30.23	28.75	75.75%	77.26%	82.86%	77.26%

Table A.33: Simulation Data (2/2)

A.4.2 Approximation of Pre-Replication Error



XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	4.00	2A	T	0	0.25	0.00
	1.00	4.00		F	2	0.75	0.38
	1.00	4.00	2B	T	0	0.25	0.00
	1.00	4.00		F	4	0.75	0.75
	1.00	4.00	2C	T	0	0.25	0.00
	1.00	4.00		F	4	0.75	0.75
	1.00	4.00	2D	T	0	0.25	0.00
	1.00	4.00		F	1	0.75	0.19
							2.06

Workflow	WF1	
Configuration	tau	3
	alpha	1
Avg. #content per path		11.25
Assumption	XOR branches are distributed uniformly	
Pre-replication error	average	2.06
	worst case (2B oder 2C)	4
Pre-replication error rate	average	**18.33%**
	worst case	**35.56%**

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	4.00	2A, 2A1	T	0	0.25	0.00
	1.00	4.00		F	4	0.75	0.75
	1.00	4.00	2B, 2B1	T	0	0.25	0.00
	1.00	4.00		F	8	0.75	1.50
	1.00	4.00	2C, 3	T	0	0.25	0.00
	1.00	4.00		F	4	0.75	0.75
	1.00	4.00	2D, 3	T	0	0.25	0.00
	1.00	4.00		F	1	0.75	0.19
							3.19

Workflow	WF1	
Configuration	tau	4
	alpha	2
Avg. #content per path		11.25
Assumption	XOR branches are distributed uniformly	
Pre-replication error	average	3.19
	worst case (2B, 2B1)	8
Pre-replication error rate	average	**28.33%**
	worst case	**71.11%**

Table A.34: Approximation of Pre-Replication Error for Workflow WF1

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	3.00	1A	T	0	0.33	0.00
	1.00	3.00		F	2	0.67	0.44
	1.00	3.00	1B	T	0	0.33	0.00
	1.00	3.00		F	5	0.67	1.11
	1.00	3.00	1C	T	0	0.33	0.00
	1.00	3.00		F	3	0.67	0.67
							2.22

Workflow	WF2		
Configuration	tau	3	
	alpha	1	
Avg. #content per path		9	
Assumption	XOR branches are distributed uniformly		
Pre-replication error	average	2.22	
	worst case (1B)	5	
Pre-replication error rate	average	**24.69%**	
	worst case	**55.56%**	

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	3.00	1A, 1A1	T	0	0.33	0.00
	1.00	3.00		F	3	0.67	0.67
	1.00	3.00	1B, 2	T	0	0.33	0.00
	1.00	3.00		F	5	0.67	1.11
	1.00	3.00	1C, 1C1	T	0	0.33	0.00
	1.00	3.00		F	7	0.67	1.56
							3.33

Workflow	WF2		
Configuration	tau	4	
	alpha	2	
Avg. #content per path		9	
Assumption	XOR branches are distributed uniformly		
Pre-replication error	average	3.33	
	worst case (1C, 1C1)	7	
Pre-replication error rate	average	**37.04%**	
	worst case	**77.78%**	

Table A.35: Approximation of Pre-Replication Error for Workflow WF2

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
2	0.50	3.00	3A1A	T	0	0.33	0.00
	0.50	3.00		F	3	0.67	0.33
	0.50	3.00	3A1B	T	0	0.33	0.00
	0.50	3.00		F	1	0.67	0.11
	0.50	3.00	3A1C	T	0	0.33	0.00
	0.50	3.00		F	2	0.67	0.22
							0.67

Workflow	WF3	
Configuration	tau	3
	alpha	1
Avg. #content per path		16.5
Assumption	XOR branches are distributed uniformly	

Pre-replication error	average	0.67
	worst case (3A1A)	3

Pre-replication error rate	average	4.04%
	worst case	18.18%

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	2.00	3A	T	0	0.50	0.00
	1.00	2.00		F	1	0.50	0.25
	1.00	2.00	3B	T	0	0.50	0.00
	1.00	2.00		F	6	0.50	1.50
2	0.50	3.00	3CA, 3A2	T	0	0.33	0.00
	0.50	3.00		F	3	0.67	0.33
	0.50	3.00	3CA, 3A2	T	0	0.33	0.00
	0.50	3.00		F	1	0.67	0.11
	0.50	3.00	3CB, 3A2	T	0	0.33	0.00
	0.50	3.00		F	2	0.67	0.22
							2.42

Workflow	WF3	
Configuration	tau	4
	alpha	2
Avg. #content per path		16.5
Assumption	XOR branches are distributed uniformly	

Pre-replication error	average	2.42
	worst case (3B, 3A1A)	9

Pre-replication error rate	average	14.65%
	worst case	54.55%

Table A.36: Approximation of Pre-Replication Error for Workflow WF3

Workflow	WF4	
Configuration	tau	3
	alpha	1
Avg. #content per path		16.8
Assumption	XOR branches are distributed uniformly	

Pre-replication error	average	3.10
	worst case	9
	(2A, 2B1C)	

Pre-replication error rate	average	18.44%
	worst case	53.57%

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	2.00	2A	T	0	0.50	0.00
	1.00	2.00		F	3	0.50	0.75
	1.00	2.00	2B	T	0	0.50	0.00
	1.00	2.00		F	2	0.50	0.50
2	0.50	2.00	2A1A	T	0	0.50	0.00
	0.50	2.00		F	1	0.50	0.13
	0.50	2.00	2A1B	T	0	0.50	0.00
	0.50	2.00		F	4	0.50	0.50
3	0.50	3.00	2B1A	T	0	0.33	0.00
	0.50	3.00		F	1	0.67	0.11
	0.50	3.00	2B1B	T	0	0.33	0.00
	0.50	3.00		F	4	0.67	0.44
	0.50	3.00	2B1C	T	0	0.33	0.00
	0.50	3.00		F	6	0.67	0.67
							3.10

Workflow	WF4	
Configuration	tau	4
	alpha	2
Avg. #content per path		16.8
Assumption	XOR branches are distributed uniformly	

Pre-replication error	average	3.96
	worst case	10
	(2A, 2A1, 2B1C)	

Pre-replication error rate	average	23.56%
	worst case	59.52%

XOR	Prob	#Path	Pre-Replication Path	Pre-Replication Success	Pre-Replication Error (#content)	Probability	Result
1	1.00	2.00	2A, 2A1	T	0	0.50	0.00
	1.00	2.00		F	4	0.50	1.00
	1.00	2.00	2B, 2B1	T	0	0.50	0.00
	1.00	2.00		F	3	0.50	0.75
2	0.50	2.00	2A1A, 2A1A1	T	0	0.50	0.00
	0.50	2.00		F	2	0.50	0.25
	0.50	2.00	2A1B, 2A1B1	T	0	0.50	0.00
	0.50	2.00		F	5	0.50	0.63
3	0.50	3.00	2B1A, 2B1A1	T	0	0.33	0.00
	0.50	3.00		F	2	0.67	0.22
	0.50	3.00	2B1B, 3	T	0	0.33	0.00
	0.50	3.00		F	4	0.67	0.44
	0.50	3.00	2B1C, 3	T	0	0.33	0.00
	0.50	3.00		F	6	0.67	0.67
							3.96

Table A.37: Approximation of Pre-Replication Error for Workflow WF4

A.4.3 Approximation of the Increase of the Transient Replica Ratio

Note that the following calculations of the approximated increase of the transient replica ratio for the workflows WF2, WF3, and WF4 used in the evaluation scenario abstracts from the order in which content needs of activities of paths other than the most probable path are scheduled. Thus, a simplified "top-down" approach is used based on numbering of the activities.

Workflow	**WF2**	
Configuration	tau	3
	alpha	1
	beta	2
Avg. #content per path		9
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	3.33
	probability	33%
	actual	1.10

Delta transient replica ratio	**12.22%**

Comment
* tau limits additional activity to be pre-replicated to a single activity

XOR	Prob	MPP	Add. Paths	Add. #Content	Avg. Add. #Content
1	1	1A	1B	5	
			1C	3	4.00
	1	1B	1A	2	
			1C	3	2.50
	1	1C	1A	2	
			1B	5	3.50
					3.33

Workflow	**WF2**	
Configuration	tau	4
	alpha	2
	beta	3
Avg. #content per path		9
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	4.44
	probability	33%
	actual	1.47

Delta transient replica ratio	**16.30%**

Comment
* tau limits additional activity to be pre-replicated to two activities

XOR	Prob	MPP	Add. Paths	Add. #Content	Avg. Add. #Content
1	1	1A	1B	5	
			1C	3	
			1B, 1C	8	5.33
	1	1B	1A	2	
			1C	3	
			1A, 1C	5	3.33
	1	1C	1A	2	
			1B	5	
			1A, 1B	7	4.67
					4.44

Figure A.58: Approximation of the Transient Replica Ratio Increase of PA / CPA over MPP for Workflow WF2

Workflow	**WF3**	
Configuration	tau	3
	alpha	1
	beta	2
Avg. #content per path		16.5
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	0.00
	probability	33%
	actual	0.00

Delta transient replica ratio	**0.00%**

Comment
1. XOR
* no add. pre-replication because activity 2 followed by XOR (=> no re-scheduling)
2. XOR
* tau limits activity to be pre-replicated to most probable path
* there is no re-scheduling because schedule is cleared after XOR

Workflow	**WF3**	
Configuration	tau	4
	alpha	2
	beta	3
Avg. #content per path		16.5
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	1.00
	probability	33%
	actual	0.33

Delta transient replica ratio	**2.00%**

Comment
1. XOR
* tau limits activity to be pre-replicated after XOR to most probable path
2. XOR
* tau limits additional activity to be pre-replicated to a single activity

XOR	Prob	MPP	Add. Paths	Add. #Content	Avg. Add. #Content
2	0.5	3A1A	3A1B	1	
			3A1C	2	0.75
	0.5	3A1B	3A1A	3	
			3A1C	2	1.25
	0.5	3A1C	3A1A	3	
			3A1B	1	1.00
					1.00

Figure A.59: Approximation of the Transient Replica Ratio Increase of PA / CPA over MPP for Workflow WF3

Workflow	**WF4**	
Configuration	tau	3
	alpha	1
	beta	2
Avg. #content per path		16.8
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	0.00
	probability	33%
	actual	0.00

Delta transient replica ratio	**0.00%**

Comment
For all XOR transitions
* tau limits activity to be pre-replicated to most probable path
* there is no re-scheduling because schedule is cleared after XOR

Workflow	**WF4**	
Configuration	tau	4
	alpha	2
	beta	3
Avg. #content per path		16.8
Assumption	XOR branches are distributed uniformly	

Avg. add. pre-replication	theoretical	5.58
	probability	33%
	actual	1.84

Delta transient replica ratio	**10.97%**

Comment
For all XOR transitions
* tau limits additional activity to be pre-replicated to a single activity

XOR	Prob	MPP	Add. Paths	Add. #Content	Avg. Add. #Content
1	1	2A	2B	2	2.00
	1	2B	2A	3	3.00
2.1	0.5	2A1A	2A1B	4	
		2A1B	2A1A	1	1.25
2.2	0.5	2B1A	2B1B	4	
			2B1C	6	2.50
	0.5	2B1B	2B1A	1	
			2B1C	6	1.75
	0.5	2B1C	2B1A	1	
			2B1B	4	1.25
					5.58

Figure A.60: Approximation of the Transient Replica Ratio Increase of PA / CPA over MPP for Workflow WF4

Chapter B
Glossary

Content

Content is an umbrella term covering knowledge as well as other product- and user-related information.

Context

"The context of a smart product comprises all aspects of the ambiance, which can be detected and reacted upon by the smart product (relevant information), such as characteristics of the user (her location, task at hand, etc.) and interfering physical or virtual properties (noise level, nearby resources, etc.). We only refer to those aspects of an ambiance as context that are not mandatory for the operation of a smart product (auxiliary information), user input, e.g., is not part of the context of a smart product." [SMVU11, p.15]

Destination

A node targeted by a message is referred to as destination. This node can be specified either explicitly (e.g., by means of the node's IP address) or implicitly. For example, structured P2P overlay network typically use node identifiers for implicit specification of message destinations. Note that if the destination cannot serve the request encapsulated in the message, it may forward the message and – in retrospect – take over the role of an intermediate.

Direct Request

All local requests are referred to as direct requests. Local requests originate from applications and / or services that are deployed on a node that operates an instance of the distributed storage service for request handling. Amongst others, this includes *PUT / GET* requests resulting from *PUT / GET* events in the simulation framework and requests made by the node's workflow engine to serve activity-related content needs.

Environment

[SMVU11] denotes environments as ambiances yet states that both terms can be used interchangeably. For reasons of simplicity, this document uses the common term *environment* according to the following definition. "An ambiance is an identifiable container with a clear boundary that may contain smart products and other, non-smart product entities and which can be attributed with certain well-known properties. Entities inside the container can influence each other but they are not influenced by anything outside the container." [SMVU11, p.14]

Erasure Code Replication

Erasure code replication strategies divide a content object c into a number of b equally-sized blocks. Moreover, erasure coding is applied to these blocks in order to generate a number of $k > b$ blocks of the same size. Hence, there is a so-called stretch factor Ω so that $k = \Omega \times b$. These k blocks are placed on k distinct nodes and *any* b out of the k distributed blocks are sufficient for reassembling and accessing the content c. Erasure code replication conforms to partial replication by setting b to a value of 1. Hence, there is no loss in generality [LCL04, LYC07]. Erasure code replication is often used to enhance content persistence and availability properties [KWZ+00, DBEN07].

Forward

Message forwarding is a special case of message sending. It is used if an intermediate, which has been assumed playing the role of destination, has to forward (i.e., send) a message to the next assumed destination, because it cannot serve the request. In short, message forwarding denotes sending of a message from a node other than the source of the message.

Indirect Request

Requests that are delivered to nodes other than the requester / initiator are referred to as indirect requests regardless of whether they can be served by the receiving node. Amongst others, this includes *PUT / GET* requests being encapsulated in related *PUT / GET* messages. Indirect requests are always based on direct requests. For example, a *GET* request resulting from a *GET* event in the simulation framework is initially handled by the local node (i.e., direct request). Only if this node cannot serve the request locally, it sends / routes the request to a potential provider. From the perspective of the latter, the request is seen as an indirect request (i.e., a requests originating from another node).

Initiator

A node that dispatches a *PUT* request in order to initially inject content into the distributed storage system is referred to as initiator.

Intermediate

A node on the delivery path from source to destination is referred to as intermediate. An intermediate receives messages either from source or other intermediates and routes / forwards messages to the destination or other intermediates.

Last Hop

A node on the delivery path of a message that routed / sent / forwarded the message to a certain node is referred to as last hop without specifying whether it plays the role of source or intermediate. Simply put, the last hop is the sender of a message.

Leaf Set

The leaf set comprises nodes that are neighbours to the local node in the logical space of the overlay network. Hence, the leaf set consists of nodes with identifiers being numerically close to the local node's identifier. For example, the leaf set of Pastry comprises $|L|$ nodes with $|\frac{L}{2}|$ nodes having smaller identifiers (i.e., counter clockwise positioned in the hash space) and the remaining $|\frac{L}{2}|$ nodes having greater identifiers (i.e., clockwise positioned in the hash space) [RD01a].

Life Cycle

[SMVU11] defines the life cycle of smart products consisting of the three successive phases *design*, *manufacturing*, and *use*. Moreover, the latter is interrelated with a fourth phase *maintenance* that can passed through alternately / if needed during product use.

Neighbour Set

The neighbour set denotes a state table of the content location and routing substrate Pastry that comprises nodes that are close according to a certain proximity metric (e.g., geographic distance). The term is a synonym for the term neighbourhood set used in [RD01a]. The term must not be confused with neighbour set used as part of the Common API, which provides a set of nodes that are neighbours to the local node in the logical space of the overlay network [DZD+03]. This set of nodes is referred to as leaf set. A generic description of the term is presented in Section 2.3.1

Node

The term node is used for all entities that play a role in the P2P overlay network and utilise the distributed storage service.

Pre-Replication

In contrast to traditional replication which is based on events occurred in past periods, pre-replication bases its decision on predictions of upcoming demand

/ access patterns (i.e., active replication). Hence, the approach is comparable with pre-fetching / pre-staging. However, while the latter results in pre-fetched / pre-staged content being cached by requesters, pre-replication distributes transient replicas.

Primary Provider

The role primary provider is a special case of the provider role. While the latter captures all nodes that store and provide content of any type, the primary provider is only played by nodes that initially store a content object after it was injected into the distributed storage system by an initiator. Hence, the role primary providers covers content objects only; replicas always result from subsequent operations.

Provider

A node storing content of any type on-board (i.e., in its shared storage) is called a provider of this content. Such nodes provide and contribute a set of content objects to the distributed storage system and are capable of serving corresponding *GET* requests.

Replica

A replica is a copy of an object.

Requester

A requester is a node that queries content by means of a *GET* request. It can also play the role of a provider in case it stores the requested content on-board, i.e., in case its*GET* request is served locally.

Route

Message delivery from source to destination by means of the actual routing mechanism of the overlay network implementation such as Plaxton's prefix matching approach [PRR99] is referred to as message routing. Note that in contrast to message sending, message routing is handled based on the destination's identifier.

Send

Message delivery from source to destination by means of direct addressing of the destination is referred to as message sending (e.g., based on a node's IP address).

Smart Product

"A smart product is a hybrid entity, consisting of a physical object and a software object. It is capable of self-organised embedding into different environments over the course of its life-cycle. It uses resources, such as storage, input or output capabilities and knowledge, which are built-in or provided by the environment, with the goal of natural and purposeful product-to-human interaction. At the same

time, it provides built-in resources and knowledge in the environment for other smart products to use (product-to-product interaction)." [SMVU11, p.11]

Source

A node that initially creates and dispatches a message is referred to as source of the message. It routes / sends messages to the destination or other intermediates.

Transient Replica

Transient replicas are not permanently stored in the network and are assigned a configurable gradually increasing TTL. The TTL value is increased each time the replica is accessed. If the access rate exceeds a certain configurable threshold, the replica is transformed into a persistent replica (i.e., transient replica becomes replica). If, otherwise, a transient replica is not accessed during the defined TTL period, it is either removed or flagged to be removed depending on the applied storage strategy.

Bibliography

[AG12] SAP AG. Energieschub für Ihre Workflows. SAP information sheet – SAP work manager by syclo, SAP AG, 2012.

[AKM07] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. MundoCore: A light-weight infrastructure for pervasive computing. *Pervasive Mob. Comput.*, 3(4):332–361, August 2007.

[AL01] Ahmed Amer and Darrell D.E. Long. Noah: Low-cost file access prediction through pairs. In *IEEE International Conference on Performance, Computing, and Communications*, pages 27–33. IEEE, 2001.

[ALA+08] Jordi Pujol Ahulló, Pedro García López, Marc Sànchez Artigas, Marcel Arrufat Arias, Gerard París Aixalà, and Max Bruchmann. PlanetSim: An extensible framework for overlay network and services simulations. Technical Report DEIM-RR-08-002, Architecture and Telematic Services Research Group, 2008.

[ALPB02] A. Amer, D.D.E. Long, J.-F. Paris, and R.C. Burns. File access prediction with adjustable accuracy. In *IEEE International Conference on Performance, Computing, and Communications*, pages 131–140, 2002.

[AMAM04] Wagner M. Aioffi, Geraldo R. Mateus, Jussara M. Almeida, and Daniel S. Mendes. Mobile dynamic content distribution networks. In *Proceedings of the 7th ACM International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 87–94, Venice, Italy, 2004.

[ASD12] Tehmina Amjad, Muhammad Sher, and Ali Daud. A survey of dynamic replication strategies for improving data availability in data grids. *Future Gener. Comput. Syst.*, 28(2):337–349, February 2012.

[ATS04] Stephanos Androutsellis-Theotokis and Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys*, 36(4):335–371, December 2004.

[BA99] A.L. Barabási and R. Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[BB04] S. Buchholz and T. Buchholz. Replica placement in adaptive content distribution networks. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 1705–1710, 2004.

[BC09] Shishir Bharathi and Ann Chervenak. Data staging strategies and their impact on the execution of scientific workflows. In *Proceedings of the 2nd International Workshop on Data-aware Distributed Computing*, New York, USA, 2009.

[BCP08] Chiara Boldrini, Marco Conti, and Andrea Passarella. ContentPlace: social-aware data dissemination in opportunistic networks. In *Proceedings of the 11th International Symposium on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, pages 203–210, 2008.

[BCW08] Wayne C. Booth, Gregory G. Colomb, and Joseph M. Williams. *The Craft of Research, Third Edition*. University Of Chicago Press, April 2008.

[Bec11] Matthias Beckerle. D4.2.2: Final concept for security and privacy of proactive knowledge. Deliverable D4.2.2, EU FP7 SmartProducts, 2011.

[BF11] João Barreto and Paulo Ferreira. Data replication support for collaboration in mobile and ubiquitous computing environments. In *Handbook of Research on Mobility and Computing*. 2011.

[BFS87] P. Bratley, B.L. Fox, and L.E. Schrage. *A guide to simulation*. Springer, 1987.

[BLMHS+13] Ralph Barthel, Kerstin Leder Mackley, Andrew Hudson-Smith, Angelina Karpovich, Martin Jode, and Chris Speed. An internet of old things as an augmented memory system. *Personal Ubiquitous Comput.*, 17(2):321–333, February 2013.

[Bot09] Maarten Botterman. Internet of things: an early reality of the future internet. Workshop report, European Commission, May 2009.

[Bra04] Karl S. Brandt. *Using multiple experts to perform file prediction*. PhD thesis, University of California, Santa Cruz, 2004.

[Bre00] Eric A. Brewer. Towards robust distributed systems. In *Proceedings of the 19th annual ACM Symposium on Principles of Distributed Computing*, Portland, Oregon, United States, 2000.

[Bro10] Marc Brogle. *IP Multicast using Quality of Service enabled Overlay Networks*. PhD thesis, University of Bern, Bern, 2010.

[Bus96] Arnold H. Buss. Modeling with event graphs. In *Proceedings of the 28th Conference on Winter Simulation*, pages 153–160, Coronado, California, United States, 1996.

[BWYH06] Ali Bahrami, Changzhou Wang, Jun Yuan, and Anne Hunt. The workflow based architecture for mobile information access in occasionally connected computing. In *IEEE International Conference on Services Computing*, pages 406–413, Chicago, IL, USA, 2006.

[CBPS10] Bernadette Charron-Bost, Fernando Pedone, and Andre Schiper, editors. *Replication : Theory and Practice*, volume 5959 of *Lecture Notes in Computer Science*. 1st edition, 2010.

[CDL+07] Ann Chervenak, Ewa Deelman, Miron Livny, Mei-Hui Su, Rob Schuler, Shishir Bharathi, Gaurang Mehta, and Karan Vahi. Data placement for scientific applications in distributed environments. In *8th IEEE/ACM International Conference on Grid Computing*, pages 267–274, Austin, TX, USA, 2007.

[CFZ01] Mitch Cherniack, Mike Franklin, and Stan Zdonik. Data management for pervasive computing. In *VLDB*, Rome, Italy, 2001.

[CGFZ03] M. Cherniack, E. F Galvez, M. J Franklin, and S. Zdonik. Profile-driven cache management. In *19th International Conference on Data Engineering, 2003*, pages 645– 656. IEEE, March 2003.

[CI97] Pei Cao and Sandy Irani. Cost-aware WWW proxy caching algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems*, Berkeley, CA, USA, 1997.

[CMR05] Miguel Castro, Manuel Miguel, and Antony Rowstron. Debunking some myths about structured and unstructured overlays. In *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2*, pages 85–98, Berkeley, CA, USA, 2005.

[Coa99] Workflow Management Coalition. Workflow management coalition – terminology & glossary. Technical Report WFMC-TC-1011, Workflow Management Coalition, 1999.

[Com09] European Commission. Internet of things. Strategic research roadmap, European Commission, 2009.

[CRBS03] Yatin Chawathe, Sylvia Ratnasamy, Lee Breslau, and Scott Shenker. Making gnutella-like p2p systems scalable. In *Proceedings of the 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 407–418, Karlsruhe, Germany, 2003.

[CSWH01] Ian Clarke, Oskar Sandberg, Brandon Wiley, and Theodore W. Hong. Freenet: a distributed anonymous information storage and retrieval system. In *International Workshop on Designing Privacy enhancing Technologies: Design Issues in Anonymity and Unobservability*, pages 46–66, 2001.

[DBEN07] Alessandro Duminuco, Ernst Biersack, and Taoufik En-Najjary. Proactive replication in distributed storage systems using machine availability estimation. In *Proceedings of the 2007 ACM CoNEXT Conference*, pages 1–12, New York, USA, 2007.

[DC08] Ewa Deelman and Ann Chervenak. Data management challenges of data-intensive scientific workflows. In *Proceedings of the 8th IEEE International Symposium on Cluster Computing and the Grid*, pages 687–692, Washington, DC, USA, 2008.

[DHJ+07] Giuseppe DeCandia, Deniz Hastorun, Madan Jampani, Gunavardhan Kakulapati, Avinash Lakshman, Alex Pilchin, Swaminathan Sivasubramanian, Peter Vosshall, and Werner Vogels. Dynamo: amazon's highly available key-value store. In *Proceedings of 21st ACM SIGOPS Symposium on Operating Systems Principles*, pages 205–220, Stevenson, Washington, USA, 2007.

[Dij59] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271, December 1959.

[DMG+11] Aba-Sah Dadzie, Alex McLean, Marina Giordanino, Daniel Schreiber, Zahid Ali Syed, Lena Vildjiounaite, Ilkka Niskanen, and Marcus Staender. D5.4.3: Final implementation of MMUI to interact with smart products & proactive knowledge. Deliverable D5.4.3, EU FP7 SmartProducts, 2011.

[DR02] P. Druschel and A. Rowstron. PAST: a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 8th Workshop on Hot Topics in Operating Systems*, pages 75–80, Elmau, Germany, 2002.

[DVDA04] J. Dehnert and W.M.P. Van Der Aalst. Bridging the gap between business models and workflow specifications. *International Journal of Cooperative Information Systems*, 13(03):289–332, 2004.

[DZD+03] Frank Dabek, Ben Zhao, Peter Druschel, John Kubiatowicz, and Ion Stoica. Towards a common API for structured peer-to-peer overlays. In *2nd International Workshop on Peer-to-Peer Systems*, pages 33–44, Berkeley, CA, USA, 2003.

[Fle10] Elgar Fleisch. What is the internet of things? an economic perspective. *Economics, Management, and Financial Markets*, 2010(2):125–157, 2010.

[Fra01] Michael Franklin. Challenges in ubiquitous data management. In *Informatics: 10 Years Back, 10 Years Ahead*, pages 24–33, 2001.

[fSidI13] Bundesamt für Sicherheit in der Informationstechnik. Cloud computing grundlagen, 2013. `https://www.bsi.bund.de/DE/Themen/CloudComputing/Grundlagen/Grundlagen_node.html`.

[Gat09] Ginger Gatling. *Workflow-Management mit SAP*. SAP PRESS, 2nd edition, 2009.

[Gee02] Dennis Geels. Data replication in OceanStore. Technical Report UCB/CSD-02-1217, Computer Science Division (EECS), University of California, Berkeley, CA, USA, 2002.

[GGL03] Sanjay Ghemawat, Howard Gobioff, and Shun-Tak Leung. The google file system. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 29–43, Bolton Landing, NY, USA, 2003.

[GHOS96] Jim Gray, Pat Helland, Patrick O'Neil, and Dennis Shasha. The dangers of replication and a solution. *SIGMOD Rec.*, 25(2):173–182, June 1996.

[GKM+11] Christine Grimm, Oliver Kasten, Markus Miche, Holger Ridinger, Patrick Real, Ivan Delchev, Elena Vildjiounaite, Victoria Uren, Andriy Nikolov, Daniel Schreiber, Matthias Beckerle, and Melanie Hartmann. D1.1.2: Final requirements for smart products and proactive knowledge. Deliverable (confidential) D1.1.2, EU FP7 SmartProducts, 2011.

[HA04] Chengdu Huang and T. Abdelzaher. Towards content distribution networks with latency guarantees. In *12th IEEE International Workshop on Quality of Service*, pages 181–192, 2004.

[Had12] Aristotelis Hadjakos. Hardware requirements for smart products. SmartProducts whitepaper, EU FP7 SmartProducts, February 2012.

[Hei07] Andreas Heinemann. Opportunistic networks. In *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 190–210. 2007.

[HG10] Pascale Hugues and Jerome Golenzer. A virtual plane to build and maintain real ones. SmartProducts whitepaper, EU FP7 SmartProducts, 2010.

[HS07] Melanie Hartmann and Daniel Schreiber. Prediction algorithms for user actions. In *Lernen - Wissen - Adaption*, pages 349–354, Halle, Germany, 2007.

[HSU11] M. Hartmann, M. Stander, and V. Uren. Adapting workflows to intelligent environments. In *7th International Conference on Intelligent Environments*, pages 9 –16, July 2011.

[HUV10] Melanie Hartmann, Victoria Uren, and Elena Vildjiounaite. Gathering knowledge for supporting interaction with smart products. In *Proceedings of the 2010 ACM Conference on Ubiquitous Computing*, pages 1–2, Copenhagen, 2010.

[JIR+09] F. Mary Magdalene Jane, N. Ilayaraja, M. Ashwin Raghav, R. Nadarajan, and Safar Maytham. Cache prefetch and replacement with dual valid scopes for location dependent data in mobile environments. In *Proceedings of the 11th International Conference on Information Integration and Web-based Applications & Services*, pages 364–371, New York, USA, 2009.

[Kö01] Frank Köster. *Analyse von Simulationsmodellen mit Methoden des Knowledge Discovery in Databases*. PhD thesis, Carl von Ossietzky Universität Oldenburg, Dept. of Computer Science, 2001.

[Kan07] Jussi Kangasharju. Peer-to-peer systems. In *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 172–189. 2007.

[KGB+11] Alexander Kröner, Patrick Gebhard, Boris Brandherm, Benjamin Weyl, Jörg Preissinger, Carsten Magerkurth, and Selcuk Anilmis. Personal shopping support from digital product memories. In *Proceedings of the 1st International Conference on Pervasive and Embedded Computing and Communication Systems*, pages 64–73, Vilamoura, Algarve, Portugal, 2011.

[KHB+11] Alexander Kröner, Jens Haupert, Boris Brandherm, Markus Miche, and Ralph Barthel. Towards a model of object memory links. In *Proceedings of the 2011 International Workshop on Networking and Object Memories for the Internet of Things*, pages 17–18, New York, USA, 2011.

[KIS11] Leyli Mohammad Khanli, Ayaz Isazadeh, and Tahmuras N. Shishavan. PHFS: A dynamic replication method, to decrease access latency in the multi-tier data grid. *Future Generation Computer Systems*, 27(3):233–244, 2011.

[KK04] M. Karlsson and C. Karamanolis. Choosing replica placement heuristics for wide-area systems. In *24th International Conference on Distributed Computing Systems*, pages 350–359, 2004.

[KKM02] Magnus Karlsson, Christos Karamanolis, and Mallik Mahalingam. A framework for evaluating replica placement algorithms. Technical Report HPL-2002-219, HP Laboratories, Palo Alto, 2002.

[KKREM04] Holger Kirchner, Reto Krummenacher, Thomas Risse, and David Edwards-May. A location-aware prefetching mechanism. In *Proceedings of the 4th International Network Conference*, pages 453–460, Plymouth, UK, 2004.

[KM02] Tor Klingberg and Raphael Manfredi. Gnutella protocol development - RFC 0.6 draft, 2002.

[KMS+12] Oliver Kasten, Markus Miche, Daniel Schreiber, Melanie Hartmann, Aristotelis Hadjakos, Pascale Hugues, Victoria Uren, Aba-Sah Dadzie, Julia Kantorovitch, Elena Vildjiounaite, Ilkka Niskanen, Julien Etienne Mascolo, Steven Luitjens, and Andriy Nikolov. D12.1.3: Rolling report on use cases and trials – final version. Deliverable D12.1.3, EU FP7 Smart-Products, 2012.

[KOS04] A. Künzer, F. Ohmann, and L. Schmidt. Antizipative Modellierung des Benutzerverhaltens mit Hilfe von Aktionsvorhersage-Algorithmen. *MMI-Interaktiv*, 7:61–83, 2004.

[KRT06] J. Kangasharju, K.W. Ross, and D.A. Turner. Adaptive content management in structured P2P communities. In *Proceedings of the 1st International Con-*

ference on Scalable Information Systems, pages 1–10, Hong Kong, 2006.

[KRT07] J. Kangasharju, K.W. Ross, and D.A. Turner. Optimizing file availability in peer-to-peer content distribution. In *26th IEEE International Conference on Computer Communications*, pages 1973–1981, 2007.

[KWZ+00] John Kubiatowicz, Chris Wells, Ben Zhao, David Bindel, Yan Chen, Steven Czerwinski, Patrick Eaton, Dennis Geels, Ramakrishna Gummadi, Sean Rhea, and Hakim Weatherspoon. OceanStore: an architecture for global-scale persistent storage. In *Proceedings of the 9th International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 190–201, Cambridge, Massachusetts, United States, 2000.

[KZL07] Christian P. Kunze, Sonja Zaplata, and Winfried Lamersdorf. Mobile processes: Enhancing cooperation in distributed mobile environments. *Journal of Computers*, 2(1), 2007.

[Law06] Averill M. Law. *Simulation Modeling and Analysis*. Mcgraw-Hill Professional, 4th edition, 2006.

[LCL04] W.K. Lin, D.M. Chiu, and Y.B. Lee. Erasure code replication revisited. In *Proceedings of the 4th International Conference on Peer-to-Peer Computing*, pages 90–97, Zurich, Switzerland, 2004.

[LCP+05] Eng Keong Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *IEEE Communications Surveys & Tutorials*, 7(2):72–93, 2005.

[LD97] Hui Lei and Dan Duchamp. An analytical approach to file prefetching. In *Proceedings of the USENIX Annual Technical Conference*, Berkeley, CA, USA, 1997.

[LXL08] Zhenyu Li, Gaogang Xie, and Zhongcheng Li. Efficient and scalable consistency maintenance for heterogeneous peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems*, 19(12):1695–1708, 2008.

[LYC07] W.K. Lin, C. Ye, and D.M. Chiu. Decentralized replication algorithms for improving file availability in P2P networks. In *15th IEEE International Workshop on Quality of Service*, pages 29–37, 2007.

[Mü07a] Max Mühlhäuser. Scalability: Two issues of global scale. In *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 21–37. 2007.

[Mü07b] Max Mühlhäuser. Smart products: An introduction. In *Constructing Ambient Intelligence*, volume 11, pages 158–164, Darmstadt, Germany, 2007.

[Mat89] F. Mattern. Virtual time and global states of distributed systems. *Parallel and Distributed Algorithms*, page 215–226, 1989.

[MBGB11] Markus Miche, Kai Baumann, Jerome Golenzer, and Marc Brogle. A simulation model for evaluating distributed storage services for smart product systems. In *8th International ICST Conference on Mobile and Ubiquitous Systems*, Copenhagen, 2011.

[MEA+11] Markus Miche, Valentin Erlenbusch, Carlo Allocca, Andriy Nikolov, Julien Etienne Mascolo, and Jerome Golenzer. D4.1.3: Final concept for storing, distributing, and maintaining proactive knowledge securely. Deliverable D4.1.3, EU FP7 SmartProducts, 2011.

[Mey13] Sonja Meyer. Internet of things-aware process modeling: Integrating IoT devices as business process resources. In *Proceedings of the 25th International Conference on Advanced Information Systems Engineering*, Valencia, Spain, 2013.

[MF10] Friedemann Mattern and Christian Flörkemeier. Vom Internet der Computer zum Internet der Dinge. *Informatik-Spektrum*, 33(2):107–121, 2010.

[MG12] Peter Mell and Timothy Grance. *The NIST Definition of Cloud Computing*. Number 800-145 in Recommendations of the National Institute of Standards and Technology. U.S. Department of Commerce, 2012.

[MKH11] Markus Miche, Alexander Kröner, and Jens Haupert. Classification of storage frameworks for digital objects memories (unpublished). In *Workshop of the W3C Object Memory Modeling Incubator Group*, Kaiserslautern, 2011.

[MLMB01] A. Medina, A. Lakhina, I. Matta, and J. Byers. BRITE: Universal topology generation from a user's perspective. Technical Report BUCS-TR-2001, Boston University, Boston, MA, 2001.

[MN05] James W. Mickens and Brian D. Noble. Predicting node availability in peer-to-peer networks. In *Proceedings of the 2005 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 378–379, New York, USA, 2005.

[MPV06] Vidal Martins, Esther Pacitti, and Patrick Valduriez. Survey of data replication in P2P systems. INRIA Research Report 6083, HAL - CCSD, 2006.

[MSB11] Markus Miche, Marcus Ständer, and Marc Brogle. Leveraging process models to optimize content placement - an active replication strategy for smart products. In *5th ERCIM Workshop on eMobility*, pages 27–38, Vilanova i la Geltrú, Catalonia, Spain, 2011.

[MSH09] Markus Miche, Daniel Schreiber, and Melanie Hartmann. Core services for smart products. In *3rd European Workshop on Smart Products*, pages 1–4, Salzburg, 2009.

[MSMP11] Sonja Meyer, Klaus Sperner, Carsten Magerkurth, and Jacques Pasquier. Towards modeling real-world aware business processes. In *2nd International Workshop on the Web of Things (in conjunction with ninth international conference on pervasive computing)*, San Francisco, California, United States, 2011.

[MV08] Wolfgang Maass and Upkar Varshney. Preface to the focus theme section: 'smart products'. *Electron. Market.*, 18(3):211–215, August 2008.

[NdGV11] Andriy Nikolov, Mathieu d'Aquin, Marina Giordanino, and Elena Vildjiounaite. D2.1.3: Final version of the conceptual framework. Deliverable D2.1.3, EU FP7 SmartProducts, 2011.

[Noc07] Zoltán Nochta. Smart items in real time enterprises. In *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*, pages 211–228. 2007.

[NUD11] Andriy Nikolov, Victoria Uren, and Aba-Sah Dadzie. D1.3.2: Final concepts for proactive knowledge. Deliverable D1.3.2, EU FP7 SmartProducts, 2011.

[OBS+08] Atsuyuki Okabe, Barry Boots, Kokichi Sugihara, Sung Nok Chiu, and D. G. Kendall. *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams, Second Edition*. Wiley Series in Probability and Statistics. Wiley, May 2008.

[On04] Giwon On. *Quality of Availability for Widely Distributed and Replicated Content Stores*. PhD thesis, TU Darmstadt, 2004.

[PB03] Stefan Podlipnig and Laszlo Böszörmenyi. A survey of web cache replacement strategies. *ACM Comput. Surv.*, 35(4):374–398, December 2003.

[PGS93] R. Hugo Patterson, Garth A. Gibson, and M. Satyanarayanan. A status report on research in transparent informed prefetching. *SIGOPS Oper. Syst. Rev.*, 27(2):21–34, April 1993.

[PJFY04] F. Perich, A. Joshi, T. Finin, and Y. Yesha. On data management in pervasive computing environments. *IEEE Transactions on Knowledge and Data Engineering*, 16(5):621–634, 2004.

[PK08] M. Pervilä and J. Kangasharju. Performance of ajax on mobile devices: A snapshot of current progress. In *2nd Intl. Workshop on Improved Mobile User Experience*, 2008.

[Pri08] Dan Pritchett. BASE: An acid alternative. *Queue*, 6(3):48–55, 2008.

[PRR99] C. G. Plaxton, R. Rajaraman, and A.W. Richa. Accessing nearby copies of replicated objects in a distributed environment. *Theory of Computing Systems*, 32(3):241–280, 1999.

[PSMS10] P. K Patra, M. Sahu, S. Mohapatra, and R. K Samantray. File access prediction using neural networks. *IEEE Transactions on Neural Networks*, 21(6):869–882, 2010.

[RD01a] A. Rowstron and P. Druschel. Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In *Middleware*, pages 329–350, 2001.

[RD01b] Antony Rowstron and Peter Druschel. Storage management and caching in PAST, a large-scale, persistent peer-to-peer storage utility. In *Proceedings of the 18th ACM Symposium on Operating Systems Principles*, pages 188–201, Banff, Alberta, Canada, 2001.

[Res09] MAYA Research. Trillions, 2009. `http://vimeo.com/7395079`.

[RF01] Kavitha Ranganathan and Ian Foster. Identifying dynamic replication strategies for a high-performance data grid. In *Grid Computing — GRID 2001*, volume 2242, pages 75–86. 2001.

[RFH+01] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content-addressable network. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 161–172, San Diego, California, United States, 2001.

[Rig09] Wolfgang Riggert. *ECM - Enterprise Content Management - Konzepte und Techniken rund um Dokumente*. Springer-Verlag, 2009.

[RP05] Natarajan Ravichandran and Jehan-François Pâris. Making early predictions of file accesses. In *Proceedings of the 4th International Information and Telecommunication Technologies Symposium*, Florianopolis, SC, Brazil, 2005.

[Sch07] Michael Schneider. Towards a general object memory. In *Proceedings of the 1st International Workshop on Design and Integration Principles for Smart Objects*, pages 1–6, Innsbruck, Austria, 2007.

[Sch11] Daniel Schreiber. From smart environments to smart products and back. SmartProducts whitepaper, EU FP7 SmartProducts, March 2011.

[SGAM11] Daniel Schreiber, Andreas Göb, Erwin Aitenbichler, and Max Mühlhäuser. Reducing user perceived latency with a proactive prefetching middleware for mobile SOA access. *International Journal of Web Services Research*, 8(1):68–85, 2011.

[She10] Haiying Shen. IRM: Integrated file replication and consistency maintenance in P2P systems. *IEEE Transactions on Parallel and Distributed Systems*, 21(1):100–113, 2010.

[SHS12] Marcus Staender, Aristotelis Hadjakos, and Daniel Schreiber. Adaptive workflows in smart environments: combining imperative and declarative models. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, pages 1171–1174, New York, USA, 2012.

[Smi82] Alan Jay Smith. Cache memories. *ACM Computing Surveys*, 14(3):473–530, 1982.

[SMK+01] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In *Proceedings of the 2001 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 149–160, San Diego, California, United States, 2001.

[SMK+10] P. Stephan, G. Meixner, H. Koessling, F. Floerchinger, and L. Ollinger. Product-mediated communication through digital object memories in heterogeneous value chains. In *2010 IEEE International Conference on Pervasive Computing and Communications*, pages 199–207, 2010.

[SMVU11] Daniel Schreiber, Markus Miche, Elena Vildjiounaite, and Victoria Uren. D1.2.4: Final concepts for smart products. Deliverable D1.2.4, EU FP7 SmartProducts, 2011.

[SPAL04] P. Shah, J.F. Pâris, A. Amer, and D.D.E. Long. Identifying stable file access patterns. In *Proceedings of the 21st IEEE Symposium on Mass Storage Systems and Technologies*, pages 159–163, 2004.

[SS05] Yasushi Saito and Marc Shapiro. Optimistic replication. *ACM Computing Surveys*, 37(1):42–81, March 2005.

[SSV+12] Daniel Schreiber, Marcus Staender, Lena Vildjiounaite, Jani Mäntyjärvi, Vili Törmänen, Asko Ollila, Steven Luitjens, Wenzhu Zou, Pascale Hugues, and Julien Etienne Mascolo. D5.5.2: Evaluation report for final implementation. Deliverable D5.5.2, EU FP7 SmartProducts, 2012.

[SSZA+11] Marcus Staender, Daniel Schreiber, Syed Zahid Ali, Philip Webster, and Alex McLean. D5.3.2: Final version of methodology and tools for multimodal UIs based on proactive knowledge. Deliverable D5.3.2, EU FP7 SmartProducts, 2011.

[SW05] Ralf Steinmetz and Klaus Wehrle. *Peer-to-Peer Systems and Applications*. Springer-Verlag New York, Inc., 2005.

[TKL10] TKLabs. CocktailCompanion: Demonstration of the platform guiding a user, November 2010. `http://www.youtube.com/watch?feature=player_embedded&v=LNXUxGPfxqU`.

[TKL11] TKLabs. Smart coffee maker, January 2011. `http://www.youtube.com/watch?feature=player_embedded&v=WWpCDu88CrI`.

[TY14] Manghui Tu and I-Ling Yen. Distributed replica placement algorithms for correlated data. *The Journal of Supercomputing*, 68(1):245–273, 2014.

[VAK+98] Geoffrey M. Voelker, Eric J. Anderson, Tracy Kimbrel, Michael J. Feeley, Jeffrey S. Chase, Anna R. Karlin, and Henry M. Levy. Implementing cooperative prefetching and caching in a globally-managed memory system. In *Proceedings of the 1998 ACM SIGMETRICS joint International Conference on Measurement and Modeling of Computer Systems*, pages 33–43, New York, USA, 1998.

[VAKC+12] Emalayan Vairavanathan, Samer Al-Kiswany, Lauro Beltrão Costa, Zhao Zhang, Daniel S. Katz, Michael Wilde, and Matei Ripeanu. A workflow-aware storage system: An opportunity study. In *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, pages 326–334, Washington, DC, USA, 2012.

[Vog08] W. Vogels. Eventually consistent. *Communications of the ACM*, 52(1):40–44, 2008.

[WNL08] Matthias Wieland, Daniela Nicklas, and Frank Leymann. Managing technical processes using smart workflows. In *Towards a Service-Based Internet*, pages 287–298, 2008.

[WPA+03] G. A.S Whittle, J. F Paris, A. Amer, D. D.E Long, and R. Burns. Using multiple predictors to improve the accuracy of file access predictions. In *20th IEEE/11th NASA Goddard Conference on Mass Storage Systems and Technologies*, pages 230–240, 2003.

[WUS10] P. Webster, V. Uren, and M. Ständer. Shaken not stirred: mixing semantics into XPDL. In *5th International Workshop on Semantic Business Process Management, Extended Semantic Web Conference*, 2010.

[YFLS11] Gala Yadgar, Michael Factor, Kai Li, and Assaf Schuster. Management of multilevel, multiclient cache hierarchies with application hints. *ACM Trans. Comput. Syst.*, 29(2):1—51, May 2011.

[YSL+06] Chansu Yu, K.G. Shin, Ben Lee, Seung-Min Park, and Heung-Nam Kim. Node clustering in mobile peer-to-peer multihop networks. In *4th Annual IEEE International Conference on Pervasive Computing and Communications Workshops*, 2006.

[YWD10] Kan Yi, Heng Wang, and Feng Ding. Decentralized integration of task scheduling with replica placement. In *9th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, pages 332–336, Hong Kong, China, 2010.

[ZHS+04] B. Y Zhao, L. Huang, J. Stribling, S. C Rhea, A. D Joseph, and J. D Kubiatowicz. Tapestry: A resilient global-scale overlay for service deployment.

IEEE Journal on Selected Areas in Communications, 22(1):41–53, 2004.

[ZKJ01] Ben Zhao, John D. Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area location and routing. Technical Report UCB-CSD-01-1141, University of California at Berkeley, USA, 2001.

Wissenschaftlicher Werdegang des Verfassers[65]

2002–2008 Studium der Wirtschaftsinformatik an der Technischen Universität Darmstadt, Darmstadt

2008–2008 Diplomarbeit am Lehrstuhl für Wirtschaftsinformatik
Technische Universität Darmstadt, Darmstadt
"Evaluation of an Architecture Framework for Electronic Cross-Organizational Collaboration"

2008–2014 Externer Doktorand am Lehrstuhl für "Telekooperation" an der Technischen Universität Darmstadt, Darmstadt

[65] Gemäß §20 Abs. 3 der Promotionsordnung der TU Darmstadt